NIST DRAFT Special Publication 800-130
June 15, 2010

National Institute of Standards and Technology
Technology Administration
U.S. Department of Commerce

A Framework for Designing Cryptographic Key Management Systems

Elaine Barker, Dennis Branstad, Santosh Chokhani, Miles Smid

C O M P U T E R S E C U R I T Y

Abstract

This Framework for Designing Cryptographic Key Management Systems (CKMS) contains descriptions of CKMS components that should be considered by a CKMS designer and specifies requirements for the documentation of those CKMS components in the design. This Framework places documentation requirements on the CKMS design document. Thus, any CKMS, that is properly documented, could have a design document that is compliant with this Framework.

KEY WORDS: access control; cryptographic key management system; cryptographic keys; disaster recovery; framework; key management functions; key management policies; key management profile; metadata; security assessment; system testing.

Acknowledgements

The National Institute of Standards and Technology (NIST) gratefully acknowledges and appreciates contributions by [Insert List] who participated in the creation, review, and publication of this document. NIST also thanks the many contributions by the public and private sectors whose thoughtful and constructive comments improved the quality and usefulness of this publication.

Authority

This document has been developed by the National Institute of Standards and Technology (NIST) in furtherance of its statutory responsibilities under the Federal Information Security Management Act (FISMA) of 2002, Public Law 107-347.

NIST is responsible for developing standards and guidelines, including minimum requirements, for providing adequate information security for all agency operations and assets, but such standards and guidelines shall not apply to national security systems. This guideline is consistent with the requirements of the Office of Management and Budget (OMB) Circular A-130, Section 8b(3), Securing Agency Information Systems, as analyzed in A-130, Appendix IV: Analysis of Key Sections. Supplemental information is provided in A-130, Appendix III.

This Framework has been prepared for use by anyone designing a Cryptographic Key Management System or anyone interested in the components of a Cryptographic Key Management System design. Nothing in this document should be taken to contradict standards and guidelines made mandatory and binding on federal agencies by the Secretary of Commerce under statutory authority. Nor should these guidelines be interpreted as altering or superseding the existing authorities of the Secretary of Commerce, Director of the OMB, or any other federal official.

Tables of Contents and Figures

Contents

Figures

1. Introduction

This Framework for Designing Cryptographic Key Management Systems (CKMS[1]) was initiated as a part of the NIST Cryptographic Key Management (CKM) Workshop[2]. The ultimate workshop goal was to define and develop technologies and standards that provide cost-effective security to cryptographic keys that themselves are used to protect computing and information processing applications. A Framework is a description of the components (i.e., building blocks) that can be combined or used in various ways to create a "system" (e.g., a group of objects working together to perform a vital function). This Framework identifies and discusses the components of a CKMS and provides requirements for CKMS design specifications conforming to this Framework.

A Framework that specifies CKMS design requirements offers the following advantages:

a) The Framework encourages CKMS designers and others to consider the factors that make a comprehensive CKMS.
b) The Framework encourages CKMS designers and others to consider factors that if properly addressed will improve CKMS security.
c) The CKMS design task is better defined by knowing the significant elements that require specification.
d) Another compliant CKMS can be compared in a logical fashion by comparing how each CKMS meets the specified requirements.

This Framework is not intended to be a CKMS design. That task is left to the system designers. Rather, the Framework provides specification requirements using menus of options from which the designers may choose.

This Framework does not mandate requirements for a secure CKMS. The requirements of this Framework are documentation requirements placed on the CKMS design. The Framework aids the designer by providing the essential components and implementation choices that form the basis of a good CKMS design. The specific choices that ensure a secure CKMS are left to the designer or other documents.

This Framework, does not mandate, requirements for the protection of U. S. government sensitive information. NIST Standards and Recommendations are referenced in this Framework as examples only. This Framework is intended to be general enough to encompass any reasonable, complete, and well designed CKMS. NIST plans to develop a CKMS Profile that is compliant with this Framework and is also appropriate for use by US Government agencies.. The NIST CKMS Profile will cite NIST Standards and Recommendations in the areas of cryptography and information security.

Requirements for conformance to this Framework are indicated by a "**shall**" in the text. Suggestions are indicated by a "should" but are not requirements for compliance.

[1] CKMS can be either singular or plural in this document and should be read as such.
[2] Held at NIST on June 8-9, 2009.

Statements of permission or recognizing some probability are indicated by "may", and statements of capability are indicated by "can".

A conformant CKMS design **shall** meet all "**shall**" requirements in this document.

1.1 Audience

This document is intended for designers, implementers, security analysts, managers, system procurers, and users of CKMS to manage and protect keys. While some introductory material is provided to explain the Framework components and to justify the requirements, this document assumes that the reader has knowledge of the principles of key management or is able to find that information elsewhere (e.g., [SP 800-57-part1]).

1.2 Organization

Section 1 provides an introduction to the Key Management Framework and the motivation behind it.

Section 2 covers basic concepts of the Framework and provides an overview of this document.

Section 3 defines the goals of a robust CKMS.

Section 4 discusses the need for CKMS security policies.

Section 5 presents the roles and responsibilities associated with a CKMS

Section 6 covers the most critical elements of a CKMS: keys and metadata, by enumerating and defining possible key types, key metadata, key management functions, and key usage functions along with security issues and protection mechanisms.

Section 7 considers the need for interoperability

Section 8 describes security controls applicable to a typical CKMS.

Section 9 describes the security testing and assurances.

Section 10 deals with disaster recovery.

Section 11 deals with on-going security assessment of the CKMS.

Section 12 discusses other items such as cost, schedule, standards, Intellectual Property Rights, laws and regulations, and technological hurdles.

Appendix A enumerates and describes applicable standards and recommendations.

Appendix B consists of a Glossary of Terms.

2. Framework Basics

2.1 Rationale for Cryptographic Key Management

Today's information systems and the information that they contain are considered to be major assets that require protection. The information used by government and business is contained in computer systems consisting of groups of interconnected computers that make use of shared networks, often referred to as the Internet. Since the Internet is shared by diverse and often competing organizations and individuals, information systems should protect themselves and the information that they contain from unauthorized disclosure, modification and use. Even the denial of service to legitimate users is considered a significant threat. The information used by these systems requires protection when it is at rest within a protected facility, and also when it is transported from one location to another.

Cryptography is often used to protect information from unauthorized disclosure, to detect modification, and to authenticate the identities of system users. Cryptography is particularly useful when data transmission or authentication occurs over communications networks for which physical means of protection are often cost-prohibitive or even impossible to implement. Thus, cryptography is widely used when business is conducted or when sensitive information is transmitted over the Internet. Cryptography also provides a layer of protection for stored data (in addition to physical and computer security access controls) against insiders who may have physical and possibly logical (e.g., system administrator) access to, but not the authorization to know or modify, the information.

Cryptographic techniques use cryptographic keys that are managed and protected throughout their life cycles by the CKMS. Effectively-implemented cryptography can reduce the scope of the information management problem from the need to protect large amounts of information to the need to protect only keys and certain metadata (i.e., information about the key and its use, such as the algorithm with which the key is to be used, the security service applied using the key, etc.). The CKMS binds a key to its critical metadata in order to control the proper use of the key.

When designing a CKMS, the cryptographic techniques used to protect the keys managed by the CKMS should offer a level of protection (often measured in bits of security) that is infeasible to bypass by a would-be attacker. This design principle is comparable to a design principle used in building safes and vaults: the designer builds the vault to a standard that would discourage the rational attacker from attempting entry; the only way to open the safe is to open the safe door by trying possible combinations until the correct combination is selected. Similarly, the only way to decrypt previously encrypted data (without knowledge of the correct key) is to test possible keys until eventually the correct key is used to decrypt the ciphertext to obtain the correct plaintext. Just as the protection provided by a safe is dependent on the number of its possible combinations, the strength of a cryptographic algorithm is dependent on the number of possible keys.

Other means of gaining access to the contents of the safe or to the information that has been encrypted may also exist. One can drill through the safe enclosure and one can attempt to find a short-cut method to crypt-analyze the cryptographic algorithm. Also, one can attempt to steal the combination or the needed key. Safe combinations and cryptographic keys both require protection. The CKMS is designed to provide the necessary protection for keys and bound metadata.

The CKMS design **shall** specify the security strength (measured in bits of security) of the cryptographic mechanisms that are employed to protect keys and any sensitive parts of their metadata.

2.2 Framework Components and Requirements

This Framework is an organized list of components and CKMS design requirements (see Figure 1 below).

Examples of components include:

a) Goals
b) Policy
c) Key Types
d) Key Metadata
e) Key Life Cycle
f) Key Management Functions
g) Security Requirements (based on the CKMS Security Policy)
h) Interoperability Requirements

The Framework contains many possible components, but the selection of which components are to be used is left to the CKMS designer who produces the CKMS design. Not all components have to be selected for a particular CKMS.

The requirements that correspond to each component are indicated by "**shall**" in this document. A compliant CKMS design can be compared to another compliant CKMS design by comparing how the requirements are met by each design.

A compliant CKMS design **shall** make selections and provide documentation as required by the requirements in this Framework.

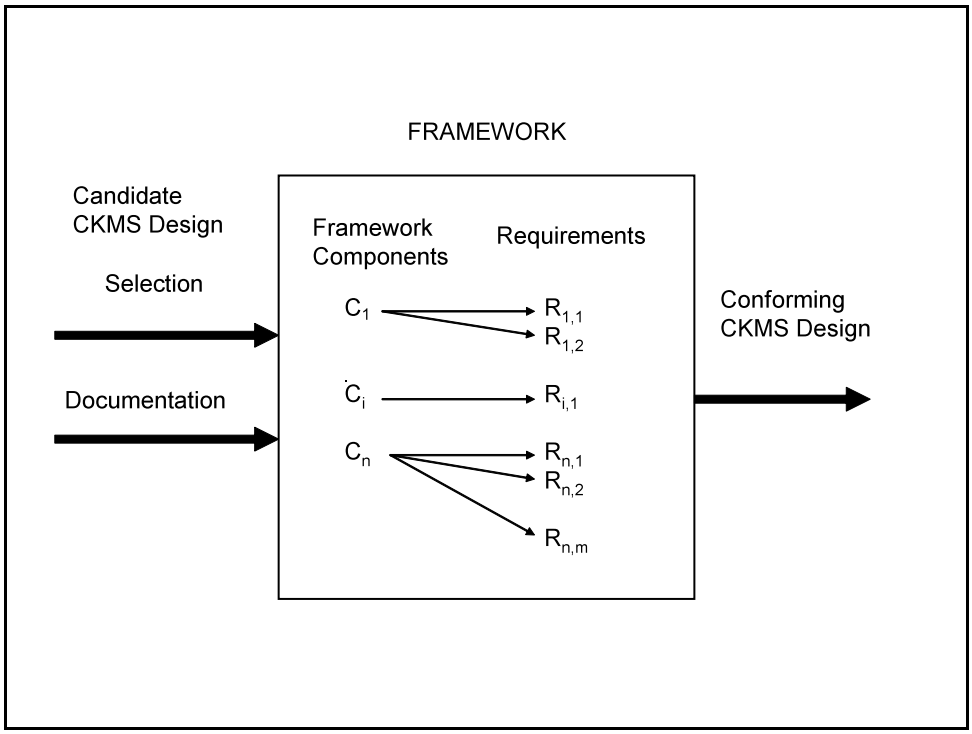

Figure 1: Framework of Components and Requirements

This document is not oriented to a particular CKMS or class of CKMS for an Enterprise or Enterprise Class (such as US Federal Government, Aerospace, Health Care, etc.). This Framework is intended to meet the needs of a wide variety of CKMS and Enterprises. This Framework may be used to develop a specific CKMS Profile for a particular organization. For example, NIST may use this Framework to develop a Federal CKMS Profile for US Federal Government (USG) agencies by selecting certain standards and protocols that comply with applicable Federal Information Processing Standards (FIPS), NIST Recommendations and guidelines (see Figure 2 below).

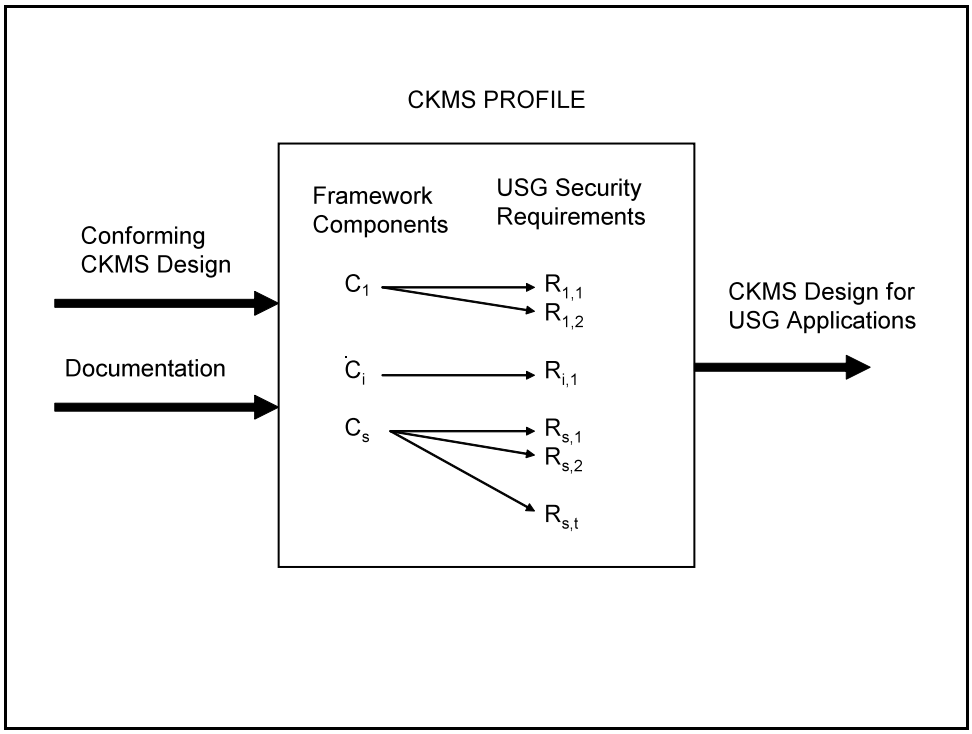

Figure 2: Federal CKMS Profile

3. Goals

A CKMS should be designed to achieve specific goals. Some possible goals are discussed in this section.

3.1 Providing Key Management to Networks, Applications, and Users

There is extensive use of cryptography in several security protocol standards (e.g., TLS, IKE, SSH, CMS, etc.) where ephemeral keys (i.e., cryptographic keys with short lifetimes that are changed often) are used by the protocols themselves. These protocols may also employ and distribute static keys (i.e., long-term keys) that are securely distributed using some other means. While, the focus of a CKMS is on the generation, distribution and storage of the static keys, a CKMS design covers the generation and storage of the ephemeral keys as well.

The network over which the CKMS operates forms the backbone of the CKMS. The CKMS designer needs to understand the efficiency and reliability of the network so that the CKMS can be designed to have minimal negative impact. The network size and scalability will provide some indication as to the number of users that the CKMS will need to handle both initially and in the future. Network characteristics such as error extension properties may also help in the selection of the cryptographic modes of operation and error detection/correction properties of the selected cryptographic algorithms that are selected.

A CKMS can be built to serve a particular application (e.g., email, data storage, healthcare systems, and payment systems) or it can be designed to serve an entire enterprise which encompasses several applications. A CKMS designed for a particular application tends to be specifically designed for that application and closely integrated into the application while an enterprise CKMS is more centralized so that common key management functions may be shared as must as possible. A CKMS designer needs to have a good understanding of the application(s) that are to be supported since they will likely affect the CKM design choices.

The CKMS designer should also study the potential users of the system. How many users will use the system for what purposes? Are the users mobile or stationary? Are the users knowledgeable of the CKMS or will it be transparent to them? Are users operating under stressful conditions where time is of the essence in getting the job done? Many CKMS designs have failed because they assumed that the user understood the purpose and importance of cryptographic keys and public key certificates. In the final analysis, if the user is hampered from doing work by the CKMS, then it will not be a successful security solution.

The goal of the CKMS designer is to specify a set of security mechanisms that function well together, provide a desired level of security that meets the needs of the application(s), are affordable, and have a minimum negative impact on operations. It is wise to consider these as well as other CKMS goals before the system is built.

The CKMS design **shall** specify its goals with respect to the communications networks on which it will function.

The CKMS design **shall** specify the applications that it will support.

The CKMS design **shall** describe anticipated number of users and the responsibilities that the CKMS places on them.

3.2 Ease of Use

While cryptography can provide very effective protection for computer information, if it is not easy to use, then it likely will not be used. A strong case can be made that the largest impediment to the implementation of cryptography is that the burden of key management is often put on the user who is either not capable of, or not willing to, perform all the security procedures required in a user-centric security system.

Ease of use provisions of a CKMS should assure that:

a) It is intuitive and easy to perform valid functions on the system (e.g., configure it or invoke a key management function);
b) It is difficult to perform invalid functions on the system;
c) It is easy to recover when a mistake is made; and
d) Recovering from a mistake is intuitive.

These rules foster a CKMS design that permits easy and effective operation, resulting in a high degree of user satisfaction. This approach also reduces the total life cycle cost by reducing the cost to support system use.

The CKMS design shall specify the functions that are performed by the various types of users.

The CKMS design shall specify any human error prevention or failsafe features designed into the system.

3.3 Workload Scalability

Performance improvements in computing and communications are major success stories in the computer industry. As performance improves, new applications require that even faster processing and communications be available. In the past, large key distribution centers often serviced a maximum of several thousand security subscribers. Now, millions of people use the Internet regularly with ever increasing demands, including new demands for keys. Demands for secure processing, data storage, and communications will continue to grow. This growth will require a CKMS to be scalable in order to meet the growing workload.

The CKMS design **shall** specify the performance characteristics of the CKMS, including average and peak workloads handled, and peak and average response times.

The CKMS design **shall** specify the extent to which the CKMS can be scaled to meet workload demands beyond peak workload. This specification **shall** be in terms of additional workload, response times for the workload, and cost.

3.4 Revocation Notification

A CKMS should have the ability to rapidly replace compromised keys (both asymmetric and symmetric) and the ability to notify the relying parties (those who make use of the key) of compromise/revocation. Compromised Key Lists (CKL), Certificate Revocation Lists (CRL) (see [RFC 5280]), White Lists, Query White Lists, and the Online Certificate Status Protocol (OCSP) (see [RFC 2560]) are examples of mechanisms in use today. Each mechanism has its benefits and drawbacks. For example, CRL and CKL have problems of growth in size and becoming out of date (i.e., staleness). Growth adversely impacts, communication, computing, and storage requirements. The growth problems for the client side can be mitigated by partitioning the revocation information into smaller chunks, each chunk handling fewer keys. Staleness cannot be fully eliminated, but can be mitigated by issuing lists more frequently. Note that in some instances, more than one revocation mechanism can be used to meet the security requirements and limitations of the relying parties.

A CKMS's key revocation notification mechanism(s) **shall** be designed based on the following considerations:

a) Relying party requirements for timeliness of revocation information;
b) Relying party computing and communication limitations; and

c) Infrastructure cost considerations.

3.5 Maximize the Use of COTS to Realize CKMS

Customers generally prefer Commercial Off-The-Shelf (COTS) products. Such products are often less costly to acquire, operate, and maintain than custom products designed and built for a single customer. However, COTS products designed and built to satisfy the "least common denominator" requirements of many customers may satisfy none of the customers. If the CKMS designer uses products that meet a range of requirements in a specific market sector, the CKMS will be more likely to be accepted in that market.

Using standard interfaces improves the extensibility of the product. Extensions and improvements should be allowed and supported by the COTS design of a CKMS so that the CKMS can be configured to meet the varying functional and workload demands, based on the number of users, transactions, keys, and application data.

The CKMS design **shall** specify the specific COTS products used in the CKMS.

The CKMS design **shall** specify which security functions are performed by COTS products.

The CKMS design **shall** specify how COTS products are configured and augmented to meet the CKMS goals.

4. Security Policies

The CKMS must be designed in a manner that supports the goals of the organization that will using the CKMS. Therefore, several policies either influence, or are dependent upon, the CKMS for protecting the organization's information. Several of these policies and their relationships are depicted in Figure 3.

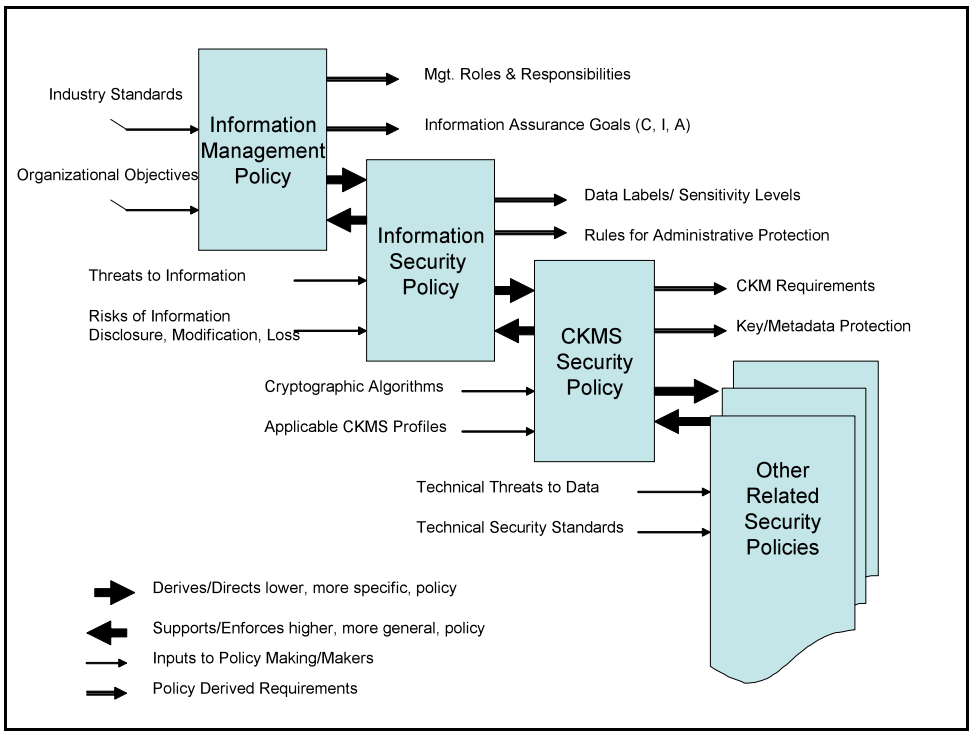

Figure 3: Related Security Policies

4.1 Information Management Policy

An organization's Information Management Policy specifies what information is be collected or created and how it is to be managed. The senior executives of an organization establish this policy using industry standards of good practices, legal requirements applicable to its information, and organizational objectives that must be achieved using the information that the organization will be collecting and creating.

The Information Management Policy typically identifies management roles and responsibilities, as well as authorities for performing these information management duties. It also specifies what information is to be considered valuable and sensitive, and how it is to be protected. In particular, this highest policy layer specifies what categories of information need to be protected against unauthorized disclosure, modification or destruction. These specifications thus form the foundation for an information security policy and dictate the levels of confidentiality, integrity, and availability protection that must be provided for various categories of sensitive and valuable information.

4.2 Information Security Policy

An Information Security Policy is usually created to support and enforce selected portions of the Information Management Policy by specifying in more detail what categories of information are to be protected from a wide spectrum of anticipated threats. The rules for collecting, protecting, and distributing valuable and sensitive information in both paper and automated form are specified in this layer of policy. The inputs to this second layer of policy include, but are not limited to, the Information Management Policy specifications, the potential threats to the security of the organization's information, and the risks of the information to unauthorized disclosure, modification, and destruction or loss.

Some of the outputs of this policy layer include the rules for administratively protecting the information by physical security means and via human security controls. Other outputs include the specification of sensitivity levels for information and specific labels for controlling access to, and protection of, the information. The Information Security Policy rules and specifications are also input to the processes involved in creating and disseminating a Security Policy for protecting the organization's automated information, especially of the highly sensitive information and the capabilities of the CKMS itself.

4.3 CKMS Security Policy

A CKMS Security Policy is created to establish and specify requirements for protecting the confidentiality, integrity, and availability of all cryptographic keys used by the organization and the metadata that is bound to each key. Inputs to this layer of policy include the selection of all cryptographic algorithms and security techniques to be used throughout the organization's automated information systems. A CKMS Profile applicable to that organization, such as a Federal CKMS Profile for all Federal agencies and Federal organizations, is also provided as one of the inputs needed for creating the CKMS security policy.

The output of this policy includes specific requirements for Cryptographic Key Management (CKM) and for the protection of all cryptographic keys and the metadata associated with, or bound to, the cryptographic keys. These requirements are used in designing, implementing, and operating the actual CKMS. Related Security Policies (e.g, the Computer Security Policy, Cryptographic Module Security Policy, etc.) must both support the CKMS Security Policy and enforce the higher layer policies for protecting the organization's information.

The CKMS Security Policy should also specify individual responsibilities and the security mechanisms to be implemented and used in order to accomplish its goals and achieve its objectives. It is essential that the CKMS Security Policy support the goals of the organization's Information Management and Information Security Policies. For example, if the Information Security Policy states that the confidentiality of the information is to be protected for up to 30 years, then the CKMS encryption algorithms and key management procedures must be selected to meet that requirement.

An organization may have different security policies covering different applications or categories of information. For example, a military-related organization may have one set of policies covering classified information and a totally different set covering personnel information. An organization may also have layered policies, with high-level policies addressing issues at the enterprise level, and lower-level policies addressing specific responsibilities and data protection methods. The high-level assignment of responsibilities may be in one document, and the detailed descriptions of how the responsibilities are to be met are placed in one or more other documents. A Physical Security Policy may be specified in one document and assigned to one organization, and a Communication Security Policy may be specified in another document and assigned to a different organization. All these policy documents may be organized in the form of a logical tree to show their relationships, and together will constitute the complete Information Management Policy for the organization.

A CKMS Security Policy must be written so that the individuals responsible for maintaining the policy can easily understand and correctly perform their roles and responsibilities. Security policies may also be written in a form (e.g., tables, formal specification languages, flow charts) such that the CKMS automatically enforces parts of the policy. Such systems may be able to check themselves for proper functioning, diagnose current or potential problems, and report the problem to the responsible party and, perhaps, automatically correct the problem.

The CKMS Security Policy for a large enterprise supporting multiple diverse organizations must accommodate the security requirements and policies of each organization. This may require the protection of data having different security levels in different security domains, and may even involve processing and storing sensitive data in "mutually suspicious" domains. Organizational Information Security Policies and the CKMS Security Policies must accommodate any allowed information sharing, and the CKMS itself must be designed to help enforce how this sharing takes place.

The CKMS design **shall** specify all types of CKMS Security Policy that it is designed to support and enforce.

The CKMS design **shall** specify the conditions under which keys and their related metadata may be shared by two or more entities and the security mechanisms that will be used to provide the protection required by the CKMS Security Policy.

The CKMS design **shall** specify how the CKMS Security Policy is to be implemented and enforced by the CKMS (e.g., the mechanisms used to provide the protection required by the policy).

The CKMS design **shall** specify the methods (e.g., tables, relational data structures, formal specification languages) to be used to express the CKMS Security Policy requirements.

The CKMS design **shall** specify how the automated portions of the CKMS Security Policy are expressed in an unambiguous tabular form or a formal language (e.g., XML or ASN.1), such that an automated security system (e.g., table driven or syntax-directed software mechanisms) in the CKMS can enforce them

5. Roles and Responsibilities

Typically, several roles are required in order to design, implement, and operate a CKMS. One person or organization can perform multiple roles, and multiple individuals may perform a single role, but a CKMS often appoints different people or organizational components to perform different roles for security and reliability purposes. Possible roles are described in the following subsections.

5.1 System Authority

A system authority is responsible to executive-level management (e.g., Chief Information Officer) for the overall operation and security of a CKMS. A system authority manages all operational CKMS roles. An operational role is a role that directly operates the CKMS.

5.2 System Administrator

System administrators are responsible for the personnel, daily operation, training, maintenance, and related management of a CKMS, other than its keys. A system administrator reports directly to the system authority.

5.3 System Designer

The CKMS designer is responsible for utilizing the CKM Framework to create a CKMS design. The designer selects the appropriate components to incorporate in the CKMS and specifies how the components will be structured, coordinated, and operated securely and efficiently.

5.4 Cryptographic Officer

A cryptographic officer is authorized to perform cryptographic initialization and management functions on the cryptographic modules. A cryptographic officer reports directly to the system authority.

5.5 Key Owner

A key owner is an entity that is authorized to use a cryptographic key or key pair. For public-private key pairs, the association is typically established through a registration process. A symmetric key may have a single, specific owner or may be shared by multiple owners.

5.6 System User

CKMS system users make use of the CKMS when key management functions are required to support an application. System users may be, and often are, key owners.

5.7 Audit Administrator

An audit administrator is responsible for auditing all aspects of a CKMS to verify its security and authorized operation. In particular, the audit administrator will manage and review the event log for all CKMS components and should have no other operational responsibilities for the CKMS. Audit administrators should not have access to any operational keys. When an audit of the cryptographic components is being performed, special audit-only keys should be created by the CKMS in accordance with the CKM policy to verify the operation of those components and then be destroyed when they are no longer needed. An audit administrator reports directly to the system authority.

5.8 Registration Agent

A registration agent is responsible for registering new entities and binding their identifiers to keys and selected metadata. The registration agent may also enter the registrant into a database of authorized entities which contains the key identifier and other bound metadata for each entity.

5.9 Key Recovery Agent

A key recovery agent is allowed to recover escrowed keys from storage after identity verification and authorization of the requesting entity is performed in accordance with the CKMS security policy.

5.10 Separation of Roles and Individuals

Multiple individuals may be assigned to each role, and a single person may have multiple roles. However, certain roles should not be shared by the same individual. For example, audit logs should be managed by someone other than a system administrator in order to detect administrative misuse or abuse. Multi-party controls (see Sections 6.7.4 and 6.7.5), should be used for highly sensitive functions. A Root Key or Master Key should be maintained under split control that assures it can only be reconstituted with the cooperation of two or more individuals.

5.11 Requirements involving Roles

The following requirements, that involve roles, are to be met by the CKMS design.

a) The CKMS design **shall** specify each role employed by the CKMS.

b) The CKMS design **shall** specify the responsibilities of each role employed by the CKMS.

c) The CKMS design **shall** specify how the individuals in each role are authenticated by the CKMS.

d) The CKMS design **shall** specify which roles require role separation.

e) The CKMS design **shall** specify how multi-person control is used for critical system functions.

f) The CKMS design **shall** specify how individual accountability is enforced.

g) The CKMS design **shall** specify the collection and storage of "audit-able" events in order to ascribe security-relevant actions to individuals or roles.

h) The CKMS design **shall** specify all automated provisions for identifying security violations, whether by individuals performing authorized roles (insiders) or by those with no authorized role (outsiders).

6. Cryptographic Keys and Metadata

6.1 Key Types

In general, cryptographic keys are categorized according to their properties and uses. Keys may be public, private, or symmetric. Keys can be either static (i.e., long term) or ephemeral (used only for a session or a single-key distribution). Various uses for a Key include signature, authentication, encryption/decryption, key wrapping, RNG, as a master, key transport, key agreement, and authorization. [SP 800-57-part1] describes twenty different key types that are shown in Figure 4 below.

Key Type
1) Private Signature Key
2) Public Signature Key
3) Symmetric Authentication Key
4) Private Authentication Key
5) Public Authentication Key
6) Symmetric Data Encryption/Decryption Key
7) Symmetric Key Wrapping Key
8) Symmetric RNG Key
9) Private RNG Key
10) Public RNG Key
11) Symmetric Master Key
12) Private Key Transport Key
13) Public Key Transport Key
14) Symmetric Key Agreement Key
15) Private Static Key Agreement Key
16) Public Static Key Agreement Key
17) Private Ephemeral Key Agreement Key
18) Public Ephemeral Key Agreement Key
19) Symmetric Authorization Key
20) Private Authorization Key
21) Public Authorization Key

Figure 4: Key Types

6.2 Key Metadata

This section lists and describes the metadata that can be bound with the various types of keys. Key metadata is defined as information associated with a particular key that specifies the secure and appropriate usage and management of the key[3]. The metadata that is appropriate for binding with a key should be selected by the CKMS designer based upon a number of factors including the key type, the key life cycle state, and the CKMS security policy. A CKMS need not bind all applicable metadata with a given key and a CKMS may not bind any metadata with some or all of the keys. See item t) below.

The following are types of metadata and their descriptions:

a) **Key Label:** A key label is a text string that provides a human-readable and perhaps machine-readable set of descriptors for the key. Examples of key labels include: "Root CA Private Key 2009-29"; "Maintenance Secret Key 2005."

b) **Key Identifier:** This text string is used by the CKMS to select a specific key from a collection of keys. A key identifier is generally unique. For public and private keys, a key identifier can be a hash value or portion of the hash value of the public key or can be CKMS assigned. The hash value is statistically unique and can be used within a specific context to make it strictly unique and thus reduce the computational overhead for the CKMS when assigning key identifiers.

c) **Key Life Cycle State:** A key life cycle state is one of a set of finite states that describe the permitted conditions of a cryptographic key. Possible states of a key include: Pre-Activation; Active; Deactivated; Compromised; Destroyed; Destroyed Compromised; and Revoked. All compromised keys should be revoked.

d) **Key Format Specifier:** This field is used to specify the format for the key and its parameters. This can be accomplished by reference to the structure using object identifiers. For example, an RSA public key consists of the modulus and a public exponent. The format specifier should specify the sequence in which these two values are stored and the format in which each value is encoded. The Internet Engineering Task Force (IETF) has defined an object identifier for storing various forms of public keys such as DSA, DH, RSA, EC, RSAPSS RSAOAEP, etc.

e) **Product used to Create the Key:** This field specifies which cryptographic product was used to create or generate the key.

f) **Cryptographic Algorithms using the Key:** This field specifies the cryptographic algorithms that can use the key. Examples include DSA, ECDSA, RSA, AES, TDES, SHA1, HMAC, etc.

g) **Schemes or Modes of Operation:** This field defines the applicable schemes, modes, or methods for performing a cryptographic algorithm while using a key.

[3] When it is understood that we are referring to the key associated with metadata, we will use the term "the key".

For example, it may specify the operation of discrete logarithm algorithms in a mathematical finite-field, binary field, or Elliptic Curve (EC) field. The schemes describe how ephemeral and static values are used to calculate a shared secret that is then used as a key or as a part of a key. For symmetric algorithms, this field may define the mode(s) of operation that can be used by the block cipher algorithm when using the key. Examples of modes of operation are Electronic Code Book (ECB), Cipher Block Chaining (CBC), Output Feedback Mode (OFB), Counter with Cipher Block Chaining-Message Authentication Mode (CCM), etc.

h) **Parameters for the Key:** This field specifies the parameters, if applicable, for a key. For example, a DSA key has the following parameters: large prime (*p*), small prime (*q*), and generator (*g*).

i) **Cryptographic Length of the Key:** This field specifies the length of the key in bits (or bytes). Examples include a 2048-bit RSA modulus, a 256-bit EC key, etc.

j) **Security Strength of the Key:** A number associated with the amount of work (that is, the base 2 logarithm of the number of operations) that is required to break a cryptographic algorithm. For example, for a TDES key of 168 bits (not including parity bits), the security strength is specified as 112 bits; for a 2048 bit RSA modulus, the security strength is specified as 112 bits.

k) **Key Type[4]:** This field identifies the key type. Key types were discussed in Section 6.1.

l) **Appropriate Applications for the Key:** This field specifies applications for which the key may be used. Examples include Kerberos, Signed E-Mail, Trusted Time Stamp, Code Signing, File Encryption, and IPSEC.

m) **Security Policies Applicable to the Key:** This field identifies the security policies applicable to the key. A key security policy for a key (as distinguished from a security policy for a CKMS) is a set of security controls that are used to protect the key during the life cycles of the key from generation to destruction (see Section 6.7). A key security policy is typically represented by an object identifier registered by the CKMS organization. Supporting different key security policies for individual keys is itself part of the CKMS security policy.

n) **Owner Identifier:** This field specifies the identifier (or identifiers) of the entity (or entities) that owns (or own) the key.

o) **Key Access Control List (ACL)[5]:** The access control list identifies the entities that can access and/or use the keys as constrained by the "access modes". This

[4] Key type also implies key usage, since usage is one of the three factors that define key type. Thus, the key usage implied by the key type should be consistent with the application of the key.

framework does not specify the access control list structure. In cases where interoperability is desired, the following items may require standardization: the syntax and semantics of the separators among access control entries; the ordering of entity and "access modes" within access control entry; the entity identifier; and the designation of bits for various "access modes".

p) **Version Number:** This field indicates the number of times the key has been changed.

q) **Parent Key:** This field points to the key from which the metadata key is derived. For example, a new key could have been derived from a TLS master secret (another key with its metadata).

This field may have two subfields:

i. **Key Identifier:** See item b) above.
ii. **Nature of Relationship:** This field identifies how the parent key is related to the child key. An example of the relationship is a mathematical function that was used to create the child key using the parent key as one of the inputs.

r) **Key Protections**[6]**:** This field specifies the integrity, confidentiality, and source authentication protections applied to the key. A public key certificate is an example of key protection whereby the CA's digital signature provides both the integrity protection and source authentication. A symmetric key and its hash value encrypted together is an example of confidentiality protection and integrity protection.

This field may have several subfields:

i. Mechanism for integrity protection (e.g., hash value, MAC, HMAC, digital signature, etc.)
ii. Mechanism for confidentiality protection (e.g., key wrap, key agreement, etc.)
iii. Mechanism for source authentication (e.g., MAC, HMAC, digital signature, etc.). Generally, a single cryptographic function (e.g., MAC, HMAC, digital signature, etc.) is used to provide integrity protection and source authentication.

[5] ACL includes authorized parties, their access mode or permission or authorization (such as create, initialize, use, entry, output, update, replace, revoke, delete, zeroize, etc.), delegation rights for each access mode, and validity period for each access mode.

[6] A key can have multiple types of protection (e.g., integrity and confidentiality). While not customary, the Framework permits multiple cryptographic mechanisms for the same security service (e.g., digital signature and MAC for integrity).

For each cryptographic mechanism for key protections, the following s**hall** be specified:

i. Cryptographic algorithm: See item f) above.
ii. Parameters for the key: See item h) above.
iii. Key identifier: See item b) above.
iv. Protection value: This field contains the protection value for integrity protection, confidentiality protection, and source authentication. For example, a MAC value or digital signature value to provide for integrity protection and/or source authentication.

s) **Metadata Protection (can be a subset of the key protections or can be different):** This field specifies the mechanisms used to provide integrity, confidentiality, and source authentication to the bound metadata. Generally, the same mechanism will be used to protect the key and its bound metadata, especially if the key and metadata are transmitted or stored together.

This field may have several subfields:

i. Mechanism for integrity protection (e.g., hash value, MAC, HMAC, digital signature, etc.)
ii. Mechanism for confidentiality protection (e.g., encryption)
iii. Mechanism for source authentication.

For each cryptographic mechanism used for metadata protection the following **shall** be specified.

i. Cryptographic algorithm
ii. Parameters for the key
iii. Key identifier
iv. Protection value (e.g., MAC, digital signature, etc.)

For the description of each of these items, see item r) above. Generally, the same mechanism will cover the key and bound metadata, especially if the key and metadata are bundled together.

t) **Metadata Binding Protection** (i.e., how the binding of metadata to the key is protected) (can be part of key protection). This requirement is implicitly satisfied if the key and bound metadata are protected as one. Otherwise, the following should be provided for the metadata binding to the key:

i. Mechanism for integrity protection (e.g., hash value, MAC, HMAC, digital signature, etc.)
ii. Mechanism for source authentication.

For each cryptographic mechanism used for metadata binding protection, the following **shall** be specified.

i. Cryptographic algorithm
ii. Parameters for the key
iii. Key identifier

iv. Protection value (e.g, MAC, digital signature, etc.)

For the description of each of these items, see item r) above). This requirement is implicitly satisfied if the key and bound metadata use a single mechanism to provide integrity protection and the same or a different single mechanism to provide source authentication.

u) **Date-Times:** The following are important date-times for life cycle state transitions of a key.
   i. Generation date (The date-time that a key was generated.)
   ii. Binding date (The date-time that a key was bound with its metadata.)
   iii. Activation date (The date-time that a key was (or can be) first used.)
   iv. Renewal date (The date-time that a key was renewed and allowed to be used for a longer period.)
   v. Rekey date (The date-time that a key was replaced with a new key that was generated so that it is completely independent of the key being replaced.)
   vi. Deactivation date (The date-time that a key was/or can be/deactivated and prevented from being used any longer.)
   vii. Expiration date (The date-time that a key's useful lifetime is terminated permanently.)
   viii. Revocation date (The date-time that a key was revoked)
   ix. Compromise date (The date-time that a key a key was known to have been compromised and marked for replacement and not renewal.)
   x. Destruction date.

v) **Revocation Reason:** If a key is revoked, this field specifies the reason for the revocation. Examples include compromise, suspected compromise, no longer assigned to the entity, misuse by the owner, etc.

For each key type used in the system, the CKMS design **shall** specify, all bound metadata elements and the circumstances under which the metadata is created and bound to the key.

For each key type, the CKMS design **shall** specify all applicable metadata elements from the list below (even if they are not bound elements):

a) Key Type
b) Cryptoperiod (for static keys)
c) Method of Generation
   i. RNG used
   ii. Key Generation Specification (e.g., [FIPS 186-3] for a DSA key, [SP 800-56A] for Diffie-Hellman key establishment)
d) For each metadata element
   i. Source of the metadata
   ii. How metadata is vetted
e) Method of Distribution

   i. Internal module key (i.e., key is created and used within the module only)
   ii. Manual
   iii. Electronic
e) Method of Key Establishment
   i. Key Transport
   ii. Key Agreement
   iii. Protocol name
f) Disclosure Protections (e.g., key confidentiality, physical security)
g) Modification Protections (e.g., HMAC or digital signature)
h) Applications that may use the key (e.g., TLS, SCL, EFS, S/MIME, IPSec, PKINIT, SSH, etc.)
i) Applications that may not use the key
j) Key Assurances
   i. Symmetric key assurances
   ii. Proof of Possession of Private Key:
      - Who performs it
      - Circumstances under which it is performed
      - How it is performed
   iii. Domain Parameter Validity Checks
      - Who performs it
      - Circumstances under which it is performed
      - How it is performed
   iv. Public Key Validity Check
      - Who performs it
      - Circumstances under which it is performed
      - How it is performed

For each key type, the CKMS design **shall** specify the protections (including binding techniques) that are applied to key and bound metadata. The CKMS design shall specify when these protections are applied and (if appropriate) when they are verified.

The CKMS design **shall** specify all syntax, semantics, and formats of all keys types and their bound metadata that will be created, stored, transmitted, processed, and otherwise managed by the CKMS.

6.3 Key Life Cycle States and Transitions

A key may pass through several states between its generation and its destruction. This section is a modification of Section 7 Key States and Transitions from [SP 800-57-part1].

6.3.1 Key States

A key is used differently depending upon its state in the life cycle. Key states are defined from a system point of view, as opposed to a single cryptographic module point of view.

a) **Pre-activation state**: The key has been generated, but is not yet authorized for use. In this state the key may only be used to perform proof of possession or key confirmation (see Section 8.1.5.1.1.2 of [SP 800-57-part1] and Section 4.2.5.5 of [SP 800-57-part1]). Other than for proof of possession or key

confirmation purposes, a key is not used to apply cryptographic protection to information (e.g., encrypt or sign information to be transmitted or stored) while in this state. Other than for proof of possession or key confirmation purposes, the key is not used to process cryptographically protected information (e.g., decrypt ciphertext or verify a digital signature) while in this state.

b) **Active state**: The key may be used to cryptographically *protect* information or to cryptographically *process* previously protected information (e.g., decrypt ciphertext or verify a digital signature) or both. When a key is active, it may be designated to protect only, process only, or both protect and process. Private signature generation keys are implicitly designated as protect only; public signature verification keys are designated as process only. A symmetric data encryption key may be used for a predetermined period of time to both encrypt and decrypt information. When that period expires, the key may transition to process only within the active state.

c) **Suspended state**: The use of a key may be suspended for a period of time. Individual modules may locally suspend the use of a key without reporting the suspension beyond the users of the module. A suspended key may be restored to an active state at a later time. A suspended key is suspended for all use unless re-activated. Eventually the suspended key is either activated or deactivated.

d) **Revoked state**: A revoked key is permanently taken out of service and will eventually be de-activated. If the integrity or secrecy of the key is suspect, the compromised key may be revoked. Revoked keys are reported in a certificate revocation list or by some equivalent mechanism. Revoked keys are typically revoked for all use. A revoked key can only transition to the deactivated state.

e) **Deactivated state**: A key whose cryptoperiod has expired but is still needed to perform cryptographic processing is deactivated until it is destroyed. A deactivated key is not used to apply cryptographic protection to information, but in some cases it may be used to process cryptographically protected information. When a key in the deactivated state is no longer required for processing cryptographically protected information, the key is destroyed.

f) **Destroyed state**: The key is destroyed so that it cannot be recovered. Even though the key no longer exists in this state, certain key attributes (e.g., key identifier, type, and cryptoperiod) may be retained.

g) **Compromised state**: Generally, keys are compromised when they are released to or determined by an unauthorized entity. A compromised key is not used to apply cryptographic protection to information. In some cases, a compromised key may be used to process cryptographically protected information, even though the confidentiality, integrity, non-repudiation or associations of the information may be suspect. For example, a compromised private signature key might be used (for integrity but not non-repudiation purposes) to verify a signature if the signed data with its signature has been

physically protected since a time before the compromise occurred. This processing is done only under very highly controlled conditions where the users of the information are fully aware of the possible consequences.

h) **Destroyed Compromised state**: The key is destroyed after a compromise, or the key is destroyed, and a compromise is later discovered. Key attributes (e.g., key identifier, type, and cryptoperiod) may be retained. This state differs from the destroyed state in that keys in this state are known, or suspected, to have been compromised.

The CKMS design **shall** specify all the states that cryptographic CKMS keys may attain.

6.3.2 Key State Transitions

Transitions between states are triggered by events, such as the expiration of a cryptoperiod, the detection of a compromise of a key, or the invocation of a key and metadata management function (see Section 6.4). Figure 5 depicts the key states and transitions.

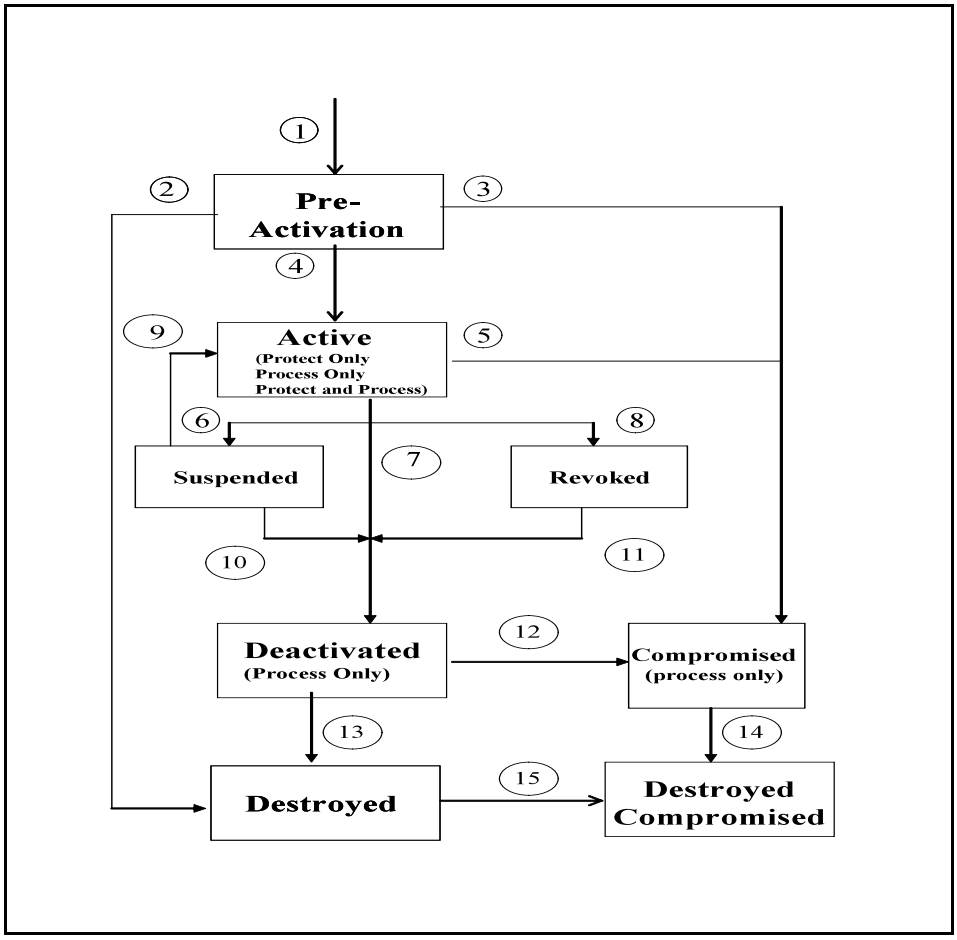

Figure 5: Key States and Transitions

Transition 1: A key enters the pre-activation state immediately upon generation.

Transition 2: A key that has never been used may transition from the pre-activation state directly to the destroyed state. In this case, the integrity of a key or the confidentiality of a key requiring confidentiality protection is considered trustworthy, but it has been determined that the key will not be needed in the future.

Transition 3: A key that is never used may transition from the pre-activation state to the compromised state when the integrity of a key or the confidentiality of a key requiring confidentiality protection becomes suspect before first use.

Transition 4: Keys transition from the pre-activation state to the active state when the key becomes available for use. This transition may be activated after reaching an activation date or by an external event. In the case where keys are generated for immediate use, this transition occurs immediately after entering the pre-activation state. This transition marks the beginning of a key's cryptoperiod.

Transition 5: An active key may transition from the active state to the compromised state when the integrity of a key or the confidentiality of a key requiring confidentiality protection becomes suspect. Generally, keys are compromised when they are released to or determined by an unauthorized entity.

Transition 6: An active key may transition to the suspended state if, for some reason, it is to be temporarily taken out of use. In this state the key is not used to protect or process data.

Transition 7: An active key may transition to the deactivated state if it is no longer to be used to apply cryptographic protection to data or no longer intended to be used to process cryptographically protected data. A key may transition from the active state to the deactivated state if the key is replaced or at the end of the key's cryptoperiod.

Transition 8: An active key may transition to the revoked state if it is determined that the key should no longer be used and all possible users should be notified of the revocation. This transition occurs with keys that are shared among entities.

Transition 9: A suspended key may transition to an active key when the reason for the suspension no longer exists.

Transition 10: A suspended key may also transition to the deactivated state if that key is no longer to be used to process data. All appropriate users should be notified that the key has been deactivated.

Transition 11: A revoked key may transition to the deactivated state. This transition may occur immediately upon revocation.

Transition 12: A deactivated key transitions from the deactivated state to the compromised state when the integrity of a key or the confidentiality of a key requiring confidentiality protection becomes suspect. Generally, keys are compromised when they are released to or determined by an unauthorized entity.

Transition 13: Assuming that a key is not determined to be compromised while in the deactivated state, a key may transition from the deactivated state to the

destroyed state. In general, a key transitions to the destroyed state as soon as it is no longer needed

Transition 14: A key in the compromised state may transition to the destroyed compromised state when the key is no longer needed to process data.

Transition 15: A destroyed key transitions to the destroyed compromised state if it is determined that the key was previously compromised. Although the key itself has already been destroyed, transition to the destroyed compromised state marks the remaining key attributes to indicate the key compromise.

The CKMS **shall** describe all transitions between the CKMS key states.

6.3.3 States and Transitions for Asymmetric Keys

The preceding discussion of key states and transitions applies to both symmetric and asymmetric keys; however, some observations that are specific to asymmetric keys are in order.

Asymmetric keys that are or will be certified are in the pre-activation state until certified or until the "not before" date specified in a certificate has passed, whichever is later. The types of transitions for asymmetric keys depend on the key type. Examples of transitions follow:

a. A private signature key is not retained in the deactivated state, but transitions immediately to the destroyed state.

b. A private signature key transitioning from the active state to the compromised state is not retained in that state, but transitions immediately to the destroyed-compromised state unless retention is required for legal purposes.

c. A public signature verification key may transition to the deactivated state at the end of the corresponding private key's cryptoperiod. The public signature verification key enters the compromised state if its integrity becomes suspect. However, public signature verification keys need not be destroyed.

d. A public key transport key transitioning from the active state is not retained in the deactivated state, but transitions immediately to the destroyed state. It enters the compromised state only when its integrity is suspect.

e. Private and public key agreement keys transitioning from the active state are not retained in the deactivated state, but transition immediately to the destroyed state.

The CKMS **shall** specify any exceptions to the key states and transitions that apply to asymmetric keys.

6.4 Key and Metadata Management Functions

The functions described in this section are performed on keys or metadata for management purposes. A CKMS should provide for the creation, modification, replacement, and destruction of keys and their bound metadata. Depending on the function, the input and/or output may have integrity, source authentication, source

authorization, and/or confidentiality services applied to them. In the case of input, the function needs to process these services. For example, for the key entry function, the calling entity may supply a secret key that is digitally signed by the source[7] and encrypted for the CKMS. The key entry function will perform the digital signature verification to authenticate the source and verify the integrity and then decrypt the encrypted key.

In the case of output, the function may need to apply security services. For example, for the key output function, the invoker may obtain a secret key that is digitally signed by the function and encrypted. The key output function will apply digital signature and encryption to the key as appropriate for the intended recipient.

The CKMS design **shall** specify the key and metadata management functions to be implemented and supported.

The CKMS design **shall** identify the integrity, confidentiality, source authentication, and source authorization services applied to each key and metadata management function implemented by the CKMS.

6.4.1 Generation

When a user requires a key, the user may request that the key be generated by the CKMS. The user may need to specify the type of key and other necessary parameters, including some metadata, when requesting this function. The function may return a key identifier that is a pointer to the key and perhaps its bound metadata. If the user wishes to actually know the key value, then the key output function (see Section 6.4.22) can be used. Keys are generated in the pre-activation state.

Key generation techniques typically depend on the specifications of the cryptographic algorithm associated with the key. Different algorithms make use of keys of differing sizes and formats. Key generation for asymmetric algorithms involves the generation of a key pair. The generation of symmetric and private keys requires the use of random number generators that are designed for key generation purposes. NIST has published several approved random number generators (see [SP 800-90]).

This function may also provide for the selection or input of metadata bound with the generated key.

The CKMS design **shall** specify the key generation methods to be used in the CKMS for each type of key.

The CKMS design **shall** specify the underlying random number generators that are used to generate symmetric and private keys.

[7] The source of the key may or may not be the calling entity of the key entry function.

6.4.2 Owner Registration

The initial registration of a security entity (i.e., individual (person), organization, device or process) and cryptographic key with bound metadata is a fundamental requirement of every CKMS. This requirement is difficult to fully automate while preserving security (i.e., protecting from the impersonation threat) and thus, it usually requires human interactions. There typically exists a registration process in a CKMS that associates each entity with an initial set of secret keys or public-private key pairs.

The CKMS design **shall** specify the process for owner registration including the process for associating keys with owners.

6.4.3 Activation

The activation function provides for the transition of a cryptographic key from the pre-activation to active state. This function may automatically activate the key. Alternatively, this function may generate a date-time metadata value that indicates when the key becomes active and can be used. A deactivation date-time may also be established using this function.

The CKMS design **shall** specify how each key type is activated.

6.4.4 Deactivation

This function transitions a key into the deactivated state. A cryptographic key is generally given a deactivation date and time when it is created and distributed. This deactivation information may be bound to the key as metadata. The period of time between activation and deactivation is generally considered the cryptoperiod of a key. This time usually has a maximum value based, in part, on the security level of the data it is protecting and the threats that could be brought against the CKMS (see [SP 800-57-part1] for further discussion). The cryptoperiod can be shortened based on the concerns of the cryptographic officer in charge of the key and data (see Section 5.4). Security policies usually state the maximum allowable cryptoperiod of any key used to protect the data covered by the policy.

The CKMS design **shall** specify how (e.g., manually or automatically based on the deactivation date-time) each key type is deactivated.

6.4.5 Revocation

A cryptographic key may be revoked at any time it is no longer authorized for use (e.g., the key has been compromised). Revoking a key is marking the key as no longer authorized for use. Security entities that have been, that are, or that will be using the key need to be notified that the key has been revoked. This may involve the publication of a revocation list identifying keys that have been revoked. Other forms of revocation notification may be supported in CKM systems.

The CKMS design **shall** specify how revocation information is made available to the relying parties.

6.4.6 Suspension

A key may be temporarily suspended[8]. Examples of situations that may warrant suspension, as opposed to irreversible revocation, include: the owner is not available for an extended period of time, the key has been misused, a possible compromise is under investigation, a token has been misplaced, etc. In addition to a security-issue related revocation (since suspension is nothing but revocation, albeit reversible), the security of re-instituting a suspended key is also critical.

The CKMS design **shall** specify how, and under what circumstances, a key can be suspended.

The CKMS design **shall** specify how suspension information is made available to the relying or communicating parties.
The CKMS design **shall** specify how a suspended key is re-activated. The CKMS design **shall** specify how the suspension function addresses source authentication, information integrity, and source authorization. The suspended key should not be used to provide security services.

The CKMS design **shall** specify how re-activation information is made available to the relying or communicating parties. The CKMS design **shall** specify how the suspension function addresses source authentication, information integrity, and source authorization.

6.4.7 Renewal

It may be desirable to have a certificate validity period that is shorter than the subject key's cryptoperiod. This reduces the size of revocation lists and revocation information, but requires certificates to be issued more frequently. Renewal permits an existing subject key to be renewed beyond its validity period.

The CKMS design **shall** specify how and the conditions under which a key can be renewed. The CKMS design **shall** specify the source authentication, integrity, and source authorization services applied to renewal requests and to renewal responses.

6.4.8 Key Update

A key can be updated by transforming it in a deterministic and synchronized manner everywhere it is needed. Key update has the possible security exposure that an adversary who obtains a predecessor key and knows the update transformation can update that (predecessor) key to the new key.

Bound metadata is updated using the modify metadata function, if available (see Section 6.4.11).

[8] Suspension is temporary deactivation. In other words, while generally deactivation is irreversible, suspension can be reversed.

The CKMS design **shall** specify all processes used to update keys and the circumstances under which the updates are allowed to take place.

6.4.9 Destruction

Keys and their bound metadata should be destroyed when they are no longer to be used. Destroying a key in a high-security application can be a complex process, depending on the storage media for the key. Historically, secure burning of paper keying material (paper tape, punched cards, or printed key lists) in a prescribed manner was used. Keys in electronic storage media may be overwritten with zeros or random patterns of zeros and ones repeatedly in a prescribed manner. Magnetic media that has a propensity for retaining low levels of magnetism may be physically destroyed, degaussed, or over-written with various bit patterns numerous times.

The CKMS design **shall** specify how and the circumstances under which a key may be destroyed (i.e., zeroized).

6.4.10 Associate Key with its Metadata

A cryptographic key may have several metadata elements bound with it. Depending on the nature of the information stored in a metadata element, the metadata element may require confidentiality protection, integrity protection, and source authentication. A modern CKMS often uses cryptography to provide this protection. Alternatively, physical protection can be provided to the key and its bound metadata so that parts of the combination cannot be replaced without authorization and the key itself cannot be disclosed to unauthorized entities.

For each key type used, the CKMS design **shall** specify how and the circumstances under which the key is bound to a set of metadata elements.

The CKMS design **shall** describe how the following security services (protections) are applied to the bound metadata and the binding of the metadata to the key: source authentication, data integrity, binding integrity, source authorization, and confidentiality.

6.4.11 Modify Metadata

The modify metadata function can be used to modify existing writable metadata that is bound with a key. Metadata that has been bound with a key should not be modifiable by an unauthorized entity without detection. For example, if the identifier of the key's owner is included in the metadata, an unauthorized entity should not be permitted to modify the key identifier without detection. Likewise, the key should not be replaceable with another key without authorization of the CKMS. The binding of a key to its metadata can be achieved using a MAC or a digital signature. The integrity of the key and its bound metadata is determined by verifying the integrity of the MAC or digital signature.

The CKMS design **shall** specify the circumstances under which bound metadata can be modified.

6.4.12 Delete Metadata

This function deletes non-required metadata bound with a key. Metadata elements may be deleted as a group, as individual elements, or as a collection of elements.

The CKMS design **shall** specify the circumstances under which the metadata bound with key can be deleted.

The CKMS design **shall** specify the technique used to delete bound metadata.

6.4.13 List Key Metadata

This function provides the ability to list bound metadata for use by an authorized user. A user may possess multiple keys and bound metadata in storage. There may be keys for digital signature generation and verification, authentication, encryption/decryption, data integrity, key establishment, and key storage

The CKMS design **shall** specify the methods to be implemented for protecting bound metadata.

6.4.14 Operational Key Storage

Operational key storage involves placing a key in local storage for use during its cryptographic period without making a copy. . Keys should be either physically or cryptographically protected when in storage (see [SP 800-57-part1]).

The CKMS design **shall** specify how, where, and the circumstances under which operational keys and their bound metadata are stored.

6.4.15 Backup Key Storage

Backup key storage involves placing a copy of a key in a safe facility so that it can be retrieved if the original is lost or modified. Backup copies of keys may be located in the same or a different facility than the operational keys to assure that the keys can be retrieved when needed even after a natural or man-made disaster.

The CKMS design **shall** specify how, where, and the circumstances under which, active keys and their bound metadata are backed up.

6.4.16 Key Archive

Key archive involves placing a key in a safe long-term storage facility so that it can be retrieved when needed. Key archiving usually requires provisions for moving the key to new storage media when the old media are no longer readable because of aging of, or technical changes to, the media readers. Archived keys should be automatically retrieved from the old storage medium and restored on the new storage medium when a storage medium replacement is made.

The CKMS design **shall** specify how, where, and the circumstances under which keys and their bound metadata are archived.

6.4.17 Key Retrieval

Obtaining a cryptographic key from storage, a backup facility, or an archive is considered retrieval if done during normal CKMS operation. If there has been an environmental or man-made disaster and the key cannot be normally retrieved and used, the key may have to be recovered by special means or with special permission (see Section 6.4.19). The CKMS security policy should state the conditions under which a key may be retrieved normally.

The CKMS design **shall** specify how, and the circumstances under which, keys and their bound metadata may be retrieved from a key database storage facility.

6.4.18 Key Escrow

Key escrow involves providing copies or components of secret or private keys to trusted parties so that the key owner or other authorized parties can recover the key when the owner's key is destroyed or otherwise unavailable.

The CKMS design **shall** specify the security policy (e.g., continuous two-person control) for the protection of escrowed keys.

The CKMS design **shall** specify how the security policy is implemented during the key escrow, i.e., how the confidentiality and multi-party control requirements are implemented during transport and storage of the escrowed key.

6.4.19 Key Recovery

Key recovery involves obtaining a copy of a key that has been previously escrowed. The key can be recovered by its owner or by an authorized third party after all the rules for recovery have been fulfilled and verified. Key recovery is normally used under circumstances where key retrieval is not possible.

The CKMS design **shall** specify the key recovery policy.

The CMSM design **shall** specify the mechanisms used to implement and enforce the key recovery policy.

6.4.20 Key Establishment

Key establishment is the process by which a key is securely created and shared between two or more entities. The key may be transported from one entity to another (key transport) or the key may be derived from information shared by the entities (key agreement). The method of transporting keys or sharing information may be either manual (e.g., sent by courier) or automated (e.g., send over the internet).

The CKMS design **shall** specify how, and the circumstances under which, keys and their bound metadata can be established.

6.4.21 Key Entry

The key entry function enters one or more keys and bound metadata into a cryptographic module in preparation for active use. A key and metadata may or may not be encrypted when entered.

The CKMS design **shall** specify how, and the circumstances under which, keys and their bound metadata can be entered into the cryptographic module.

The CKMS design **shall** specify how the integrity and confidentiality (if necessary) of the entered keys and bound metadata are protected and validated upon entry.

6.4.22 Key Output

The key output function outputs one or more keys and bound metadata out of a cryptographic module for external use or storage. Output may be for archive, backup, escrow, or normal, operational purposes A module that serves as a key generation facility may output keys for subsequent distribution. A key and its bound metadata may or may not be encrypted when output.

The CKMS design **shall** specify how, and the circumstances under which, keys and their bound metadata can be output from a cryptographic module.

The CKMS design **shall** specify how the confidentiality and integrity of the output keys and bound metadata are protected.

If a private key, symmetric key, or confidential metadata is output in plaintext form, the CKMS design **shall** specify how the calling entity is authenticated before the key is provided.

6.4.23 Validate Domain Parameters

This function performs certain validity checks on the public domain parameters associated with a public key algorithm. Passing these tests provides some assurance that the domain parameters are arithmetically correct and secure (see [SP 800-89] and [SP 800-56A]).

The CKMS design **shall** specify how, where, and the circumstances under which, public domain parameters are validated.

6.4.24 Validate Public Key

This function performs certain validity checks on a public key to provide some assurance that it is arithmetically correct. These tests typically depend on the public-key algorithm for which the key is intended, but do not depend on knowledge of the private key (see [SP 800-89], [SP 800-56A], and [SP 800-56B]).

,

The CKMS design **shall** specify how, where, and the circumstances under which, public keys are validated.

6.4.25 Validate Public Key Certification Path

This function validates the certification path (also known as a certificate chain), from the trust anchor of the relying party to a public key that the relying party needs to establish trust in. Validation of the certification path gives assurance that the subject identity given in the certificate is, in fact, the owner of the static public key and the holder of its associated static private key (assuming that proof of private key possession was verified by the certificate authority).

The CKMS design **shall** specify how, where, and the circumstances under which, a key certification path can be validated.

6.4.26 Validate Symmetric Key

This function performs certain tests on the symmetric key and its bound metadata. These tests might involve checking for the proper length and format of expected parameters. This command may also verify any error detection/correction codes or integrity checks placed upon the key and its bound metadata.

The CKMS design **shall** specify how, where, and the circumstances under which, symmetric keys and their bound metadata are validated.

6.4.27 Validate Private Key (or Key Pair)

This function performs certain tests on the private key to give assurance that it meets its specifications. Since this involves the private key, the test can only be performed by the private key owner or a trusted party acting on behalf of the private key owner. This test may also involve a pair-wise consistency test which verifies that the private key performs a complementary function to the public key. For example, in the case of an RSA key pair, applying the private key to a given input block followed by applying the public key to the result should always yield the given input block (see Section 6.4.1 of [SP 800-56B] for more information).

The CKMS design **shall** specify how, where and the circumstances under which, private keys or key pairs and their bound metadata can be validated.

6.4.28 Validate Possession of Private Key

This function is used by an entity that receives a public key and wishes to verify that the claimed owner of the public key is in possession of the corresponding private key. The sending party is typically required to perform a function with the private key that can be verified using the public key. For example, the sender of the public key may sign the public key and other information. The receiver uses the public key to validate the signature on the sent data (see [SP 800-56A], [SP 800-56B], and [SP 800-89]).

The CKMS design **shall** specify how, and the circumstances under which, possession of private keys and their bound metadata can be validated.

6.4.29 Perform Cryptographic Function using the Key

The main key usage functions are the actual functions that provide the cryptographic protection to data. For example, these functions include digital signature calculation, digital signature verification, encryption, decryption, key wrapping, MAC calculation, and MAC verification. They are performed within a cryptographic module.

The CKMS design **shall** specify all cryptographic functions that are supported.

6.4.30 Manage Trust Anchor Store

A CKMS using or providing public key certificates will require that relying parties have one or more trusted public keys. These public keys are also referred to as trust anchors. A trust anchor is required in order to establish trust in other public keys that are not otherwise trusted. The trust in these otherwise un-trusted public keys is established by verifying all signatures on a chain of public key certificates (termed "certification path" in Section 6.4.25) starting with a trust anchor trusted by the relying party. Thus, the integrity of trust anchors is critical to the security of the CKMS. The CKMS typically supports trust anchor management functions, such as add, delete and store.

The CKMS design **shall** specify how the trusted anchors are securely promulgated so that the relying parties can perform source authentication, source authorization and integrity verification on the promulgated trust anchors.

The CKMS design **shall** specify how the trust anchors are managed in relying party systems to ensure that only authorized additions, modifications, and deletions are made to the relying party system's trust anchor store.

6.5 Cryptographic Key and Metadata Security: In Storage

When cryptographic keys are submitted for storage, they are typically submitted with some metadata. The metadata may include an owner identifier or user access control list. If any of this metadata is incorrect, then the false information will be perpetuated by the CKMS system. Therefore, a CKMS key storage system should verify the authorization of the submitting entity and the integrity of the submitted data before any data is stored[9].

The CKMS design **shall** specify the methods used to verify the authorization of the entity submitting keys and bound metadata for storage.

The CKMS design **shall** specify the methods used to verify the integrity of keys and metadata submitted for storage.

When cryptographic keys are stored, they require protection. Symmetric keys and private keys require confidentiality protection and access control. All keys require integrity protection. For confidentiality protection, cryptography, computer security, and/or physical security are employed. If symmetric key cryptography is used for key

[9] It is also a good practice to verify the integrity of keys and metadata immediately upon access and before operational use.

confidentiality, then often there exists a symmetric key-encrypting-key that is used to encrypt and decrypt the stored keys and confidential metadata. This key should be protected in some manner, e.g., using physical security, that usually does not involve encryption. If asymmetric key cryptography is used for key confidentiality, the user public key is used to encrypt stored keys. The associated private key that is used to decrypt the keys should also be protected in some manner, e.g., using physical security, that usually does not involve encryption.

The CKMS design **shall** specify the methods used to protect the confidentiality of symmetric and private stored keys and bound metadata.

If a key-encrypting key is used to protect stored keys, then the CKMS design **shall** specify the methods used to protect the key-encrypting key and control its use.

All keys require integrity protection, because a garbled key will not correctly perform its intended function. Physical security can provide integrity protection for keys, but additional methods are frequently used. An error detecting code can detect an unintentional garble to a key, and an error correction code can correct certain garbles. However, if a key could be intentionally garbled, then a cryptographic integrity check like a MAC or digital signature should be implemented for error detection. If an uncorrectable garble is detected, the garbled key cannot be used. When public keys are contained within a certificate, they are provided integrity protection by means of the digital signature on the certificate. If public keys are stored outside of their certificate, then their integrity needs to be protected by some other means.

The CKMS design **shall** specify the methods used to protect the integrity of stored keys and bound metadata.

A CKMS should only allow authorized users to have access to stored symmetric and private keys. Thus, a CKMS should have some type of access control system (ACS). The ACS may be as simple as requiring a password or cryptographic key from the authorized user of the key, or it may make use of biometric authentication techniques.

The CKMS design **shall** specify how access to stored symmetric and private keys is controlled.

A key may be garbled, lost, or destroyed to the extent that it cannot be recovered by error correcting codes. If the key is a symmetric key or a private decryption key, this could result in the loss of the data protected by the key. A CKMS should employ methods for backing-up, archiving, and recovering keys as necessary to provide for the recovery of valuable data. Appendix B of [SP 800-57-part1] provides guidance on the recovery procedures for various key types.

A garble in key metadata could result in the misuse of the key or the denial of service. Therefore, metadata may also require backup, archiving, and recovery.

The CKMS design **shall** specify the techniques used for recovering any keys and metadata stored in backup or an archive.

6.6 Cryptographic Key and Metadata Security: During Key Establishment

Keys and metadata can be established between entities wishing to communicate using either key transport or key agreement methods. These methods are typically used to establish keys over automated communications networks but they could also be used to provide extra security (beyond physical protection) when keys are manually distributed. When keys are transported, one entity establishes the key to be shared, and the key along with bound metadata are transported to the other party. When keys are agreed upon, both parties contribute information that is used to derive a shared key. [SP 800-56A] and [SP 800-56B] specify cryptographic schemes for key establishment.

6.6.1 Key Transport

When cryptographic keys and metadata are transported (distributed) from one secure location (data sender) to another (intended data receiver), they should be protected. Symmetric keys and private keys require confidentiality protection and access control. All keys require integrity protection. For confidentiality protection, either physical security or cryptography is used. A manually distributed key could be physically protected by a trusted courier, or a physically protected channel could be used. Very often, the keys are sent electronically over networks that are susceptible to data eavesdropping and modification. If cryptography is used to protect the confidentiality of symmetric and private keys during transport, then a key establishment technique involving either a symmetric key-wrapping-key or, one or more asymmetric key-transport-key pairs is used. These wrapping and transport keys also should be protected by the end entities involved in the transport.

The CKMS design **shall** specify the methods used to protect the confidentiality of symmetric and private keys during their transport.

All transported keys require integrity protection because a garbled key will not correctly perform its intended function. Physical security can provide integrity protection for keys, but often other methods are used, due to the lack of physical protection of electronic data on typical networks. An error detecting code can detect an unintentional garble to a key, and an error correction code can correct certain garbles. However, if a key could be intentionally garbled, then a cryptographic integrity check like a MAC or digital signature should be used for error detection. If an uncorrectable garble is detected, a new or corrected key should be established before use. When public keys are contained within a certificate, they are provided integrity protection by the digital signature on the certificate.

The CKMS design **shall** specify the methods used to protect the integrity of transported keys and how they are implemented to recover from detected errors.

The receiver of a transported key wants assurance that the key came from an authorized key sender. This assurance is typically provided by the use of a cryptographic mechanism that authenticates the identity of the sender to the receiver.

The CKMS design **shall** specify if/how the identity of the key sender is authenticated to the receiver of transported keying material.

6.6.2 Key Agreement

Two entities, working together, can create and agree on a cryptographic key without the key being transported from one to the other. Each entity supplies some information that is used to derive a common key, but an eavesdropper obtaining this information is not able to determine the agreed-upon key. This is known as key agreement. Cryptographic algorithms employing key-agreement keys are used by each entity.

The CKMS design **shall** specify each key-agreement scheme supported by the CKMS.

Each entity participating in a key agreement needs assurance as to the identity of the other entity. This assurance is typically provided by the use of a cryptographic authentication mechanism.

The CKMS design **shall** specify if/how the identifiers of the initiator and responder in a key agreement are assured to each other entity.

6.6.3 Key Confirmation

When keys are established between two entities, each entity may wish to have confirmation that the other party did in fact establish the correct key. Key confirmation schemes are used to provide this capability. [SP 800-56A] and [SP 800-56B] specify key confirmation schemes for use in Federal CKMS. Other methods may also be appropriate.

The CKMS design **shall** specify each key confirmation method used to confirm that the correct key was established with the other party.

6.6.4 Key Establishment Protocols

Several protocols have been developed for the provision of cryptographic keys for both storage and transmission. Often these protocols are designed for a particular application or set of applications. Some well-known key establishment protocols include:

a) Internet Protocol Security (IPsec)
b) Transport Layer Security (TLS)
c) Secure/Multipart Internet Mail Extensions (S/MIME)
d) Kerbero
e) Over-The-Air-Rekeying (OTAR) Key Management Messages (KMMs)
f) Domain Name Service Extensions (DNSSEC)
g) Encrypted File Systems (EFS)
h) Secure Shell (SSH)

A high-level overview of items a) through g) can be found in [SP 800-57-part3] along with guidance as to which cryptographic options are recommended for U.S. Government use. For Secure Shell, see [RFC 4251].

A CKMS design **shall** specify all the protocols that are employed by the CKMS for key establishment and storage purposes.

6.7 Key Management Function Controls

This section describes how the initiating human is identified and authenticated, how the authorizations are verified, the metadata constraints that should be met in order to perform the various state transitions, and the metadata conditions which inhibit state transitions.

The security of a CKMS depends on the proper sequence and execution of the key management functions described in Section 6.4. The execution of these functions may be driven by time, an event, a human, or some combination of these options. Therefore, an access control system is required to assure that key management functions are only performed in response to requests (calls) by authorized entities and are appropriate for the key state. The access control system can be external to the functions, or it can be incorporated into the functions themselves. Often, this involves the establishment of a secure channel between the entity making the function calls and the cryptographic module where the functions are performed.

The CKMS design **shall** specify how access to the key management functions is restricted to authorized entities.

Even if the calling entity is authorized to call a key management function, the call may be refused for some reason. For example, the metadata may indicate that the function is inappropriate under the existing conditions.

The CKMS design **shall** specify the constraints on the key management functions in order to assure proper operation.

6.7.1 The Access Control System (ACS)

Keys are sensitive security parameters that should be protected from unauthorized disclosure, modification or usage. Therefore, key management functions should be protected by an Access Control System (ACS) to ensure that only an authorized entity is permitted to execute a particular function. An ACS could be very simple; for example, any user could be authorized to perform any key management function with any key, but this is generally not the case in a real CKMS. Logically, a CKMS control system could operate as depicted in Figure 6. A function call consisting of the calling entity's identifier, the entity authenticator, the function name, the key identifier is presented to the ACS. If the ACS determines that the function is permitted for the key and bound metadata, then the ACS notifies the function logic so that the function can be performed. If, however, the ACS determines that the function should not be allowed, the ACS returns a function denied indicator.

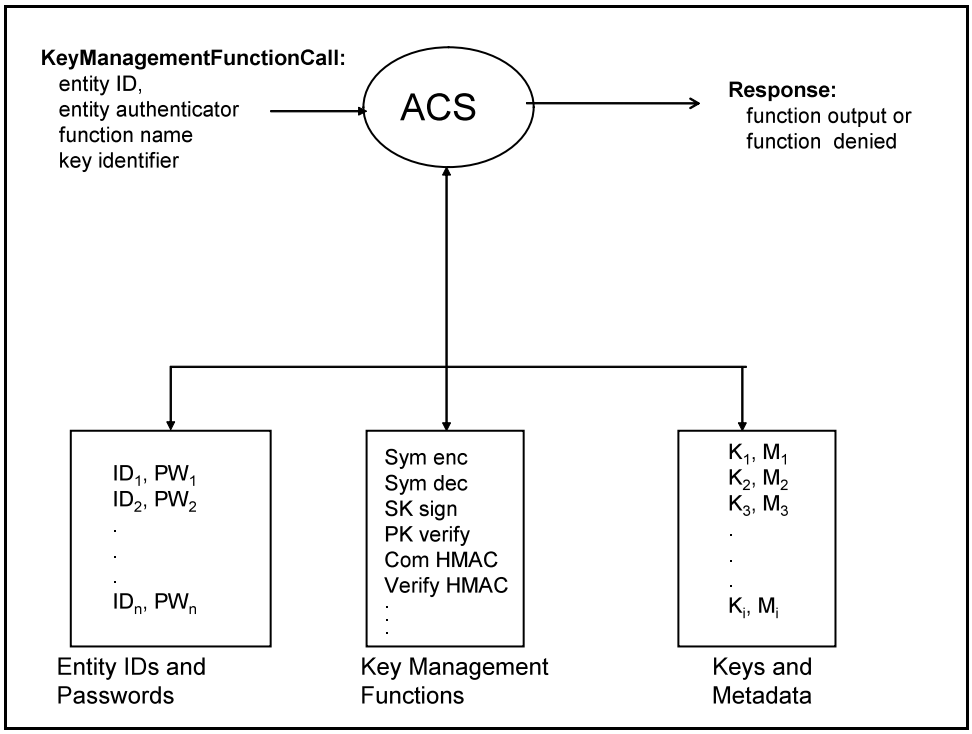

Figure 6: Sample Key Use Function Control Logic

The CKMS design **shall** specify the ACS and its associated policy for controlling access to key use functions.

The ACS makes the decision to perform the requested function or not. This decision is primarily based on the calling entity, the security policies of the CKMS, the function, the key, and its metadata. The metadata of a key plays a critical role in determining the controls that are to be enforced. For example, an organization may decide that multiple users will be permitted to use a shared key to encrypt and decrypt a particular file, while another file can be decrypted only by a single user. The CKMS policies should support and enforce the information management policies of the managing organization. Therefore, it is highly desirable that a CKMS access control system be flexible enough to accommodate various information protection policies.

The CKMS design **shall** specify the capabilities of its ACS to accommodate, implement, and enforce various information protection policies.

6.7.2 Restricting Human Access to Keys

Since symmetric and private keys are highly sensitive, a well-designed CKMS will minimize the need for the calling entity to actually be in possession of the plaintext key. This is particularly important when the calling entity is a human. A well-designed CKMS should keep these plaintext keys in physically protected devices, such as physically protected cryptographic modules. These modules may be used to generate the keys and to

perform cryptographic functions on behalf of humans so that they need never see a plaintext symmetric or private key. This feature makes the CKMS more transparent and more secure. A private key-transport key may be generated, within and never leave, the module. Keys requiring output from the module may be transported using a key transport scheme. A symmetric encryption/decryption key may then be output and transported in encrypted form using the public key of the receiving entity. Similarly, key may be securely stored outside of the module when encrypted under a public key-storage key. Sometimes, plaintext key output is permitted to support legacy systems. In such cases, multi-party control, discussed below, should be considered.

The following requirements are identical to requirements expressed in Section 6.4.21 and Section 6.4.22.

The CKMS design **shall** specify what keys (if any) may be output from the cryptographic module in plaintext form.

The CKMS design **shall** specify what keys (if any) may be entered into a cryptographic module in plaintext form.

6.7.3 Restricting Calling Entities

The calling entity (function initiator) may be a human or a device. If the key management function requires human input, then there is a dependence on the human for the accuracy and security of the input. In addition, there may be a dependency on the human to enter the input at the proper time or when the proper event occurs. In this case, the issue arises as to what action the system should take if the human input is not provided. If such functions can be performed automatically by the CKMS when they are necessary, the system becomes more transparent to the user and possibly more secure.

For each key use function, the CKMS design **shall** specify which parameters require user input, the format of the input, how the input will be processed, and what alternatives exist for the result of processing the input.

6.7.4 Multiparty Control

Certain key management functions may require multiple cooperating individuals to perform the function. This multiparty control may be enforced by requiring *k* of *n* individuals to authenticate to and be authorized by the function's access control system before the function is performed.

The CKMS design **shall** specify all multiparty control systems that are used, specifying *n* and *k* for each system.

For each multiparty system used, the CKMS design **shall** cite or specify the rationale (logic, mathematics) as to why any *k* of the *n* components will enable the desired function but *k*-1 of the components will not.

6.7.5 Key Splitting

Key splitting is a special case of multiparty control. When a highly sensitive key is required, *n* key components are generated so that any *k* of the components can be used to form the key, but any *k*-1 components provide no knowledge of the key. Each of the *n* components is then assigned to one of the *n* trusted individuals so that the key cannot be formed unless *k* of the individuals all agree to take part. If any *k*-1 of the individuals had their components compromised, the key could still not be recovered by an attacker having all the k-1 components. Thus, the security of the key is distributed. Split knowledge procedures have been used to establish root or master keys that provide protection to many other keys and whose compromise would result in a major disaster. These components are often entered into, or output from, the CKMS in plaintext form for backup purposes.

The CKMS design **shall** specify all key splitting systems that are used by specifying *n* and *k* for each system and describing how each key is generated from the k components.

For each (*n*, *k*) key splitting system used, the CKMS design **shall** specify the rationale (logic, mathematics) as to why any *k* of the *n* components can form the key, but *k*-1 of the components provide no information about the key.

6.8 Compromise Recovery

In an ideal situation, the CKMS would protect all keys and sensitive metadata so that data requiring confidentiality protection is never compromised, and data requiring integrity protection is never modified by unauthorized parties. However, since it is difficult or even impossible to design a perfect CKMS that prevents all potential security problems, a CKMS should be designed to detect compromises and unauthorized modifications, to mitigate their undesirable effects, to alert the appropriate parties of compromises, and to recover (or help recover) to a secure state once a compromise or unauthorized modification is discovered. The selection of appropriate components from this framework and levying appropriate security requirements on the selected components reduce the chance of a compromise and increase the chance of detecting a compromise or of detecting violations that could cause a compromise. This section addresses how the recovery from a compromise should occur.

When a CKMS compromise is detected

a) The compromise should be evaluated to determine its cause and scope
b) Compromise mitigation measures should be instituted to minimize the amount of data exposed
c) Appropriate corrective measures should be instituted to prevent the reoccurrence of the compromise
d) The CKMS should be returned to secure operating state.

6.8.1 Key Compromise

Depending on the key type a compromise of a key may result in

a) Loss of data confidentiality for data encrypting/decrypting keys
b) Loss of integrity for data integrity keys

c) Loss of authentication for authentication keys
d) Loss of non-repudiation and data integrity for signature keys.

Note that loss of a security service for a key is likely to result in a loss of the same and potentially other security services for data protected by the key. For example, a loss of the integrity for a public key may impact the confidentiality of the data encryption key protected by the public key and that, in turn, could compromise the confidentiality of the data protected by the data encryption key.

A key compromise may be undetected, detected or suspected. A CKMS should limit the exposure of key compromises by establishing a cryptoperiod or usage limit for each key that it uses[10]. At the end of each cryptoperiod a new key may be established to replace the old key. When a new key is established and activated to protect new data, the old key should no longer be used to protect new data. Thus, unless the compromise recurs with the new key, the new data will be protected. Of course, the old data that was protected with the old key could have been compromised, but the extent of the damage has been limited, as long as the old key did not provide any security service in replacing the new key. If a key compromise is detected, then the compromised key and other keys whose security depends upon the security of the compromised key should be replaced as soon as possible. Since the compromise of a key may result in the compromise of many other keys that it protects, it is important to design a CKMS to minimize the impact of key compromise. [SP 800-57-part1] provides guidance as to appropriate cryptoperiods for various key types.

If a key-wrapping key, a private key-transport key, or a private key-agreement key is compromised, then transported or agreed upon keys might be compromised as well. If the compromise is undetected, the compromise of additional keys might continue indefinitely. Some protocols are designed to prevent or mitigate such attacks. However it is generally considered a good idea to manually replace certain keys from time-to-time.

The CKMS design **shal**l specify the range of acceptable cryptoperiods or usage limits of each type of key used by the system.

For each key, a CKMS design **shall** specify the other key types that depend on the key for their security and how those dependent keys are to be replaced in the event of a compromise of the initial key.

The CKMS design **shall** specify the means by which other compromised keys can be determined when a key is compromised.

6.8.2 Cryptographic Module Compromise

Since a cryptographic module contains plaintext keys at some point in time, the compromise of the module has the potential to compromise the symmetric and private keys contained within the module (see Section 8.4). This could lead to the loss of

[10] The usage of keys may be limited based on a criterion such as the amount of data processed using the key or the number of times the algorithm was initialized using the key.

confidentiality, loss of integrity, or loss of authentication, as described in Section 6.8.16.8.1 above. Cryptographic modules can be compromised either physically (obtaining direct access to the keys within the module) or by non-invasive methods (obtaining knowledge of the keys within the module by some external action). To provide physical protection, modules should operate in a space to which unauthorized access is not permitted or in which unauthorized access is quickly detected before a serious compromise occurs. Some modules provide this protection at their cryptographic boundary, but larger boundaries may also be involved. [FIPS 140-2] provides requirements for the physical protection of a cryptographic module's contents. If access to the contents of a cryptographic module is permitted, then an access control system would be required to ensure that only authorized parties succeed. When a cryptographic module is compromised, the module should be excluded from normal operation until the module integrity is revalidated, and the module is rekeyed.

Following an actual or suspected cryptographic module compromise, a secure state of the module should be achieved. The actions to return to this state are collectively called recovery. Recovery may require the replacement of internal hardware and/or software of the module. The module should be returned to a secure state before the module can be returned to operation. Following repair or replacement, a module must be tested for operational capability as well as security status.

The CKMS design **shall** describe how physical access to cryptographic module contents is restricted to authorized entities.

The CKMS design **shall** specify the approach to be used to recover from a cryptographic module compromise.

To provide protection against non-invasive attacks on a cryptographic module, either the use of the module should be restricted to only trusted users, or the module should be designed to prevent this specific type of attack. In the first case, there is always the threat that a module will be lost or stolen or that a trusted user will become dishonest. In the second case, it can become very costly to protect against every possible type of non-invasive attack. An attacker might recover information about a cryptographic key used by the module by examining the power consumption of the module during the cryptographic processing. Other non-invasive attacks are based on the amount of time certain cryptographic functions take to execute, or the emanations given off by the module during its normal operation.

The CKMS design **shall** identify any modules that are not vulnerable to non-physically invasive attacks. The CKMS design **shall** provide the rationale for such exclusions.

The CKMS design **shall** describe what non-invasive attacks are mitigated by the cryptographic modules used by the system and reference a description of how the mitigation is performed.

6.8.3 Computer System Compromise Recovery

The compromise of CKMS software components or major portions of a computer operating system can be detected using tools that run on a separate secure platform and monitor any modification to a file, changes to the hash value of a file's contents, or changes to a file's attributes (e.g., owner, ACL, etc.) (See Section 8.2.4). When critical files undergo unauthorized modifications that are detected by the event log or monitoring utility, these files can be replaced using known valid and secure files located in secure storage.

If pervasive unauthorized changes to software are made, the software can undergo full recovery as described in Section 10.5.

The CKMS design **shall** describe the mechanisms used to detect unauthorized changes to the CKMS system hardware, software and data.

The CKMS design **shall** describe how the CKMS recovers from unauthorized changes to the CKMS system hardware, software and data.

6.8.4 Network Security Controls and Compromise Recovery

The scope of network security controls includes boundary devices, such as a firewall, a VPN, an intrusion detection system, and an intrusion protection system. The scope of network security controls excludes cryptographic functions, cryptographic protocols, and cryptographic services, except when used for the operation of the aforementioned network security control devices.

The following are some of the examples of compromises of network security controls:

a) The physical compromise of a network security control device.
b) A compromise of one or more cryptographic keys used by a network security control device.
c) A compromise of one or more keys used to administer the network security control device.
d) A change in the network architecture resulting in a compromise (e.g., someone connecting a VPN-connected workstation to an unsecure network and the VPN workstation being used to attack the Intranet).
e) A compromise of a privileged user (e.g., administrative) password.
f) A compromise of a platform operating system.
g) A compromise of a network security application (e.g., firewall, IDS, etc.).
h) A compromise due to a new attack on a protocol.

If physical security is compromised, the device should be replaced with a new device and physical security controls should be reviewed, repaired, and enhanced, as appropriate.

If device or administration keys are compromised, the keys should be replaced. An assessment should be conducted to determine the cause of the compromise, and corrective actions should be taken. In the unlikely event of the security strength of the

key being an issue, the key sizes may need to be increased and/or more secure cryptographic algorithms may need to be used.

If the network architecture assumptions are violated, the cause of the violation should be reviewed, and appropriate actions should be taken.

Compromised network devices should be re-validated and returned to a secure state.

If passwords are compromised, the passwords should be replaced. The users may need further training in selecting the password, in understanding password entropy, in changing passwords frequently, and in maintaining the confidentiality of written-down passwords. An examination should also be made of the authentication protocols to determine if password sniffing, online dictionary attacks or offline dictionary attacks are feasible.

If the platform operating system is compromised, one or more of the following actions should be considered, and appropriate corrective measures taken:

a) Make sure that all the latest operating system security patches are installed.
b) Ask the operating system vendor if there is a patch for the compromise.
c) Determine if a device configuration change or if blocking some protocols will prevent future attacks of the same nature as the one that caused the compromise.

If the network security application is compromised, one or more of the following actions should be considered, and appropriate corrective measures taken:

a) Make sure that all the latest network security patches are installed.
b) Ask the application vendor if there is a patch for the compromise.
c) Determine if a device change, an application configuration change, or the blocking of certain protocols will prevent future attacks that allowed or caused the compromise.

If the compromise is due to an inadequate network security protocol, one or more of the following actions should be considered, and appropriate corrective measures taken:

a) Ask the network security application vendor if there is a patch for the compromise.
b) Determine if a device configuration change or if the blocking of certain protocols will prevent future attacks of the same nature as the one that caused the compromise.

In all of these situations, the incident should be fully investigated to determine what other systems and keys may have been compromised due to a compromise of network security controls. This damage assessment can lead to additional compromise declarations and additional compromise recovery procedures.

The CKMS design **shall** specify how to recover from the compromise of the network security control used by the system. Specifically,

a) The CKMS design **shall** specify the compromise scenarios considered for each network security control device.
b) The CKMS design **shall** specify which of the mitigation techniques specified in this section were employed for each envisioned compromise scenario.
c) The CKMS design **shall** specify any additional or alternative mitigation techniques that were employed.

6.8.5 Violation of Procedures and Recovery from Violations

Security is both a technical problem with technical solutions and a human problem with procedural solutions. A CKMS should have a security policy that specifies security procedures for all roles, authorities, administrators, users, and operators. If the procedures are not correctly followed, an intentional, but unauthorized, or unintentional loss of security could result. The security policy should be understood by all personnel, each category of personnel associated with a CKMS should be tested in the roles to which they are assigned, and both the operational and recovery procedures should be practiced on a periodic basis. Some procedures can and should be enforced or verified by the automated portions of the system.

The CKMS design **shall** specify the CKMS security procedures that are to be followed by personnel role specified by the CKMS design.

For each role that is implemented, the CKMS design **shall** specify the training required for the CKMS security procedures.

The CKMS design **shall** specify how to recover from violations of CKMS security policy and procedures.

6.8.6 Personnel Compromise Recovery

A security failure is any event that compromises the secure functioning of the CKMS. A CKMS should be designed to

a) minimize the ability of humans to cause security failures,
b) determine who or what caused the security failure, and
c) mitigate the negative consequences of the failure.

Humans should be hired, cleared, trained, and monitored. The separation of duties is designed to limit the negative consequences that any single individual can cause. Multi-person control for critical functions requires the cooperation of at least two individuals to cause a security failure. Any detected security failure should result in the initiation of recovery procedures based upon the security policy.

Typical responses may include

a) the complete shut-down of the system,
b) the activation of a hot or warm backup facility and system with new keys,
c) the notification of current and potential users of the possible security failure,
d) the flagging of the keys that were compromised, or

In addition to the above responses, failures involving personnel compromise may vary from administrative reprimands, to removal from the role or position and legal action involving civil or criminal courts.

The CKMS design **shall** specify procedures and design features for recovering from the compromise of personnel security involving accidental and intentional breaches of security.

6.8.7 Physical Security Compromise Recovery

The physical security of a cryptographic module is covered in item Section 6.8.2, and the general compromise of keys is covered in Section 6.8.1. However a physical security breach of a CKMS may involve compromises other than the compromise of keys or of cryptographic modules. If security-related logic resides outside of the CKMS cryptographic modules, then the integrity of that logic also should be protected. Typically, techniques similar to those used by the cryptographic module are employed.

The CKMS design **shall** specify how the CKMS is protected from unauthorized physical access.

The CKMS design **shall** specify how the CKMS detect unauthorized physical access.

The CKMS design **shall** specify how the CKMS is recovered from unauthorized physical access to components other than cryptographic modules.

Once security is breached, the integrity of the entire breached area should be suspect. The CKMS should inform the appropriate entity as specified in the security policy of the breach so that mitigation actions can be taken. In addition, it may not be sufficient to replace all sensitive data within the breached area, because the attacker could have modified or added to the logic within the area so that the new keys and sensitive information could also be compromised in the future.

The CKMS design **shall** specify the appropriate entities to be notified of a physical security breach.

The CKMS design **shall** specify how breached areas can be re-established to a secure state.

7. Interoperability and Transition Requirements for CKMS

Interoperability is the ability of diverse systems to communicate and work together (interoperate)[11]. A CKMS may have interoperability requirements with an application or a peer CKMS. Interoperability can only be achieved by having a detailed specification to which the CKMS intends to interoperate. This inherently involves the following:

[11] See http://en.wikipedia.org/wiki/Wik/interoperability for more information on the power and uses of interoperability.

a) Common interfaces and protocols, i.e., the syntax and semantics of interfaces to invoke functions and services from one CKMS to another CKMS are the same for interoperating systems.
b) Formats for keys, metadata, and other exchanged data are the same or are understood by interoperable systems.
c) Associated data exchange mechanisms, including security mechanisms, are the same or compatible between interoperable systems.

The CKMS design **shall** specify how compatibility and interoperability requirements across component interfaces and among system components are to be satisfied.

The CKMS design **shall** specify the standards, protocols, interfaces, supporting services, commands and data formats required to interoperate with the applications it is intended to support.

The CKMS design **shall** specify the standards, protocols, interfaces, supporting services, commands and data formats required to interoperate with other CKMS for which interoperability is intended.

The CKMS design **shall** specify all external interfaces to all applications and other CKMS in order to support easy replacement or update of external components (devices, software modules).

If security is improved in one CKMS component, but the component is no longer interoperable with peer components having older security mechanisms, the new component will generally not be accepted in the marketplace. For example, if a new encryption algorithm is installed at some entity in a network, then only the entities with the new algorithm capability will be able to communicate with the new technique. Other entities will likely continue to communicate using the older algorithms. Unless accommodation is made for the smooth transition to the new algorithm (e.g., by allowing the use of legacy algorithms where necessary), the transition will be slow even when the new algorithm offers significant benefits. This is especially true because the security lifetime of a cryptographic algorithm is only an estimate and the old algorithm may actually be secure for additional years. Thus, a smooth transition may require the capability to support the use of at least two algorithms simultaneously. In that case, the cryptographic protocols should be designed to identify and negotiate which algorithm will be used in a particular key establishment transaction.

Current cryptographic algorithms should be implemented so that they can be augmented or replaced when needed. See [SP 800-57-part1] for the NIST-recommended lifetimes of government-approved cryptographic algorithms. A CKMS should only use algorithms whose security lifetime will cover the anticipated lifetime of the CKMS and the information that it protects. If the CKMS is intended to remain in service beyond the security lifetimes of its cryptographic algorithms, then there should be a transition strategy for migration to stronger algorithms in the future.

The CKMS design **shall** specify all provisions for transitions to new interoperable peer components.

The CKMS design **shall** specify any provisions made for upgrading or replacing its cryptographic algorithms.

The CKMS design **shall** specify a transition plan for upgrading and replacing cryptographic algorithms with similar, but more secure or functionally-improved algorithms.

The CKMS **shall** specify how interoperability will be supported during cryptographic algorithm transition periods.

The CKMS design **shall** specify its protocols for negotiating the use of cryptographic algorithms.

8. Security Controls

A CKMS can consist of several types of components, including but not limited to:

a) Computer Systems
b) Cryptographic Modules
c) Firewalls
d) VPN Concentrators
e) IDS and IPS

A CKMS will likely require a computer system for various functions, such as key generation, key storage and retrieval, key distribution, cryptographic module control, metadata generation, distribution and management, etc.

A CKMS should require a cryptographic module to generate, store, use and protect cryptographic keys.

A CKMS will likely be networked to distribute keys and metadata to the various users and end entities. In such situations, the CKMS should be protected using network security control devices, such as firewalls, VPN, IDS and IPS.

The following subsections of this section describe physical security control requirements and technical security control requirements for each of these CKMS component types.

8.1 Physical Security Controls

CKMS components should be physically protected in order to ensure information security. Without good physical security, the components can be tampered with and the hardware and/or software can be modified to bypass security altogether or for the circumstances selected by the attacker.

As mentioned earlier, a CKMS can have many types of components. In addition, a given CKMS may consist of a number of components of any given type. A number of

components could be required for performing different CKMS functions, to handle the CKMS workload, or to support disaster recovery. Disaster recovery is further discussed in Section 10 where hot, warm and cold standby components are discussed (see Section 10.4). All of the components of each type should have physical security protections. CKMS components will likely be located at multiple facilities]:

a) Primary Facility
    i. Operational Components
    ii. Hot Standby Components
    iii. Warm Standby Components
    iv. Cold Standby Components
    v. Backup Components
b) Secondary/Backup Facilities
    i. Additional Operational Components
    ii. Hot spare Components
    iii. Warm Spare Components
    iv. Cold Spare Components
    v. Additional Backup Components.

A CKMS can consist of one or more primary facilities and one or more backup facilities. Each of these facilities should be protected. At each facility, CKMS components can consist of active, standby or backup components, each of which should be protected.

One or more of the following mechanisms should be chosen to physically protect a CKMS, depending on the security criticality of the components. All components (regardless of type) listed above should require physical security. The following are examples of physical security mechanisms. Some of the mechanisms listed below are detection mechanisms, which should be augmented with appropriate prevention mechanisms.

a) Fences
b) Gates and doors
c) Guards
d) Locks (keyed or combination)
e) Card readers
f) Biometric devices
g) Alarm systems
h) Surveillance camera
i) Entry and exit log

The CKMS design **shall** specify the physical security protections for access to each facility holding the CKMS components.

The CKMS design **shall** specify the physical security protections implemented by the CKMS components so that they are only accessible by authorized CKMS personnel.

8.2 CKMS Component Computer Security Controls

This section addresses the computer security controls for the various CKMS components. Note that the components of a CKMS that are built upon a general-purpose operating system should also have computer security controls.

8.2.1 Operating System Security

A secure operating system is the foundation for securing a computer system. Without ensuring that the underlying operating system is secure, the security of CKMS components and the data running on the computer system cannot be assured. A secure operating system has the following security features:

a) Self-protection features to protect the operating system from modification by users and user processes.
b) Isolation features to provide and maintain separate domains of execution for the users and user processes so that they do not interfere with each other and thus compromise the security policy of data separation.
c) Access controls and operating system functions to allow users to share data based on user, group or security labels.
d) Event logging capabilities in order to support personal accountability and to investigate anomalies
e) User CKMS account management including individual identification and authentication.

The CKMS design **shall** specify all secure operating system requirements (including required operating system configurations) for the various CKMS components.

For each of the components, third party assurances based on applicable Information Security Standards (e.g., the Common Criteria, etc.) **shall** be specified by the CKMS design.

Note that CKMS components that perform dedicated security functions and do not provide a general-purpose CKMS component development, loading, or processing capability, may have reduced or minimal operating system requirements. As an example, consider a special-purpose appliance loaded with firmware and/or software to perform intrusion detection functions. This appliance may not have an operating system, and hence has no operating system security requirements. Another example is a firewall or intrusion detection system built on a "locked down" operating system so that the capability to load other CKMS components is not available.

8.2.2 Individual CKMS Component Security

A CKMS may consist of a variety of components. Each component should also be designed to protect itself from users and keep user sessions isolated. Depending on the system design and functional requirements, the CKMS component may provide finer-grained access control and CKMS component-specific event logging that is not captured by the operating system. Thus, a well-designed CKMS component should have the following security features:

a) Self-protection from other CKMS components by utilizing operating system process isolation.
b) Self-protection from CKMS component users.
c) Isolation features to provide and maintain separate sessions for the users and user processes so that they do not interfere with each other and thus violate the security policy of data separation.
d) Fine-grained access controls on CKMS component-level objects (e.g., keys and metadata or DMBS rows and tables) to allow users to share data based on user, group or security labels.
e) CKMS component level event logging in order to support personal accountability and to investigate anomalies.
f) User account management for the CKMS.

The CKMS design **shall** specify the individual security controls required for each CKMS component.

8.2.3 Malware Protection

CKMS components that receive communications, data, files, etc. over unprotected networks should scan the information for malware. Malware protection may be less critical if no information is received over unprotected networks or if all information is strongly (e.g., cryptographically) authenticated. Malware protection falls in the following three general categories:

a) Anti-virus software protects CKMS components from unwittingly installing and executing programs that perform unintended actions and may cause security compromise.

b) Anti-spyware software protects CKMS components from unauthorized parties obtaining system administrator status or authorized user status and prevents the spyware from taking on authorized CKMS component behavior.

c) Rootkit detection and prevention software protects CKMS components from rootkit malware that changes the configuration setting of the operating system in order to replace system code and hide processes and files, including the rootkit code itself, from anti-virus and anti-spyware software.

In order to be effective, malware protection should be configured for the following:

a) A daily scan,

b) A scan of removable media when first introduced,

c) A scan of newly introduced files,

d) A weekly update of the malware protection software,

e) A weekly update of the malware signature database.

The CKMS design **shall** specify the following malware protection requirements for the various CKMS components:

a) Anti-virus protection software, including specified time periods and events that trigger anti-virus scans, software update, and virus signature database updates.

b) Anti-spyware protection software, including specified time periods and events that trigger anti-spyware scans, software update, and virus signature updates.
c) Rootkit detection and protection software, including specified time periods and events that trigger rootkit detection, software update, and signature updates.

8.2.4 System Monitoring

In order to protect the integrity and confidentiality of the data files of the CKMS, system monitoring tools may be deployed. These tools execute on the platform being monitored or on another platform dedicated to monitoring various hosts. These monitoring tools can detect modifications to system files or their access control attributes and post alerts and audit events (see Section 6.8.3).

The CKMS design **shall** specify system monitoring requirements for sensitive system files to detect and/or prevent their modification or modification to their security attributes, such as access control lists.

8.3 Network Security Control Mechanisms

This section addresses the network security control mechanisms for each of the computer systems involved in the CKMS and the systems that use the keys. Examples of network security control mechanisms include:

a) Firewalls
b) Filtering Routers
c) VPN
d) Intrusion Detection Systems (IDS)
e) Intrusion Prevention Systems (IPS)
f) Adaptive Network Security Controls
    i. Adaptive Filtering
    ii. Adaptive Detection
    iii. Adaptive Prevention

Networked CKMS components are protected using a mix of firewalls and intrusion detection and prevention systems. While the firewalls provide protection by filtering out unwanted and unneeded protocols and by examining permitted protocol data to reduce the chances of attack, intrusion detection and prevention systems complement the firewalls by examining host and network activity to determine if the systems are being attacked. Thus, both firewall and intrusion detection and prevention systems should be used to properly protect intrusion detection systems.

Boundary control devices (such as firewalls, filtering routers, VPN, IDS, IPS, etc.) should be hosted on computer systems (see Section 8.2) or should be implemented in dedicated hardware devices. These devices should be placed in physically secure locations (see Section 8.1 for physical security controls). These devices should be configured with only the user accounts and network services that are required for their operation.

The CKMS design **shall** specify the boundary protection requirements for the CKMS components. These specifications, at a minimum, **shall** be in terms of the following details:

a) The types of firewalls and protocols permitted through the firewalls, including the source and destination for each type of protocol.
b) The types of intrusion detection and prevention systems used, including their logging and security breach reaction capabilities.

For each of the components, third party assurances based on applicable Information Security Standards (e.g., the Common Criteria, etc.) **shall** be specified by the CKMS design.

8.4 Cryptographic Module Engineering Controls

A cryptographic module is a set of hardware, software and/or firmware that implements security functions (e.g. cryptographic algorithms and key establishment schemes). The module encompasses everything within the cryptographic boundary[12] and includes the boundary itself.

Two security issues should be addressed regarding the security of the contents of cryptographic modules: the integrity of the security functions and the protection of the cryptographic keys and bound metadata. [FIPS 140-2] specifies requirements on cryptographic modules for maintaining the integrity of the module's security functions. However, the cryptographic modules discussed in this framework are not necessarily [FIPS 140-2]-validated cryptographic modules. Techniques such as the *software/firmware integrity test* and *known answer test*, along with physical protection from unauthorized alteration, are specified in the FIPS. Since the cryptographic keys that are associated with the algorithms are present in plaintext form for some period of time within the module, physical security measures are necessary to protect keys from unauthorized disclosure, modification, and substitution. A cryptographic boundary may be used to provide the necessary physical protection. Otherwise, physical protection is provided for a larger space that includes the cryptographic module.

Vendors of hardware cryptographic products or modules often build physical security safeguards into their devices by using strong metal cases, locks, alarms, and key zeroization mechanisms. However, software cryptographic applications running on general-purpose computers face additional risks because these computers were not designed and built to provide sufficient protection to cryptographic keys. In fact, the very computers on which the cryptography runs usually contain software written by unknown parties that have not been reviewed for security. It is therefore critical that cryptographic software be physically protected in a safe environment (i.e., one which resists tampering) and logically protected from external exploitation. [FIPS 140-2] provides some guidance and requirements regarding these protections.

[12] A cryptographic boundary is an explicitly defined perimeter that establishes the boundary of all components of a cryptographic module.

The CKMS design **shall** identify the cryptographic modules that it uses and their respective security policies.

a) The security policy of each module **shall** specify the type of module (software, hardware, or hybrid).
b) The security policy of each module **shall** specify the mechanisms used to protect the integrity of the module itself.
c) The security policy of each module **shall** specify the physical and logical mechanisms used to protect its cryptographic keys.
d) The security policy of each module **shall** specify the third party testing and validation that was performed upon the contents of the module (including security functions) and the protective measures provided by the entire module.

9. Testing and System Assurances

In this section, the term "CKMS device" may refer to any component of a CKMS or to an entire CKMS itself. A CKMS device may be composed of hardware, software, firmware or any combination thereof. A CKMS device may undergo several types of testing to ensure that it has been built to conform to its design, that it conforms to various standards, that it continues to operate according to its design, that it is interoperable with other CKMS devices, and that it can be used in larger systems for which it is intended. A CKMS device may undergo tests in the categories listed below.

9.1 Vender Testing

Device vendors test their devices to debug errors and to verify that they work as expected. The techniques and specifics of this category of testing are often considered proprietary information by the vendor and are generally not made public.

A CKMS design **shall** specify non-proprietary vendor testing that was performed on the system.

9.2 Third Party Testing

A vendor may permit a third party (other than the developer or a customer) to test its CKMS device for conformance to a particular standard. Third party testing provides confidence that the vendor did not overlook some flaw in its own testing. The National Institute of Standards and Technology (NIST) established several programs for validating conformance to its cryptographic Standards and Recommendations. In order to claim conformance with one of these Standards or Recommendations, the device should be tested by a NIST-accredited testing laboratory.

The CKMS design **shall** specify all third party testing programs passed by either the system or its devices.

9.3 Interoperability Testing

Interoperability testing, in its most general form, merely tests that two or more devices can be interconnected and operate with one another. This means that the data exchanged between the devices should be in a format that each device can use. Interoperable devices may be interconnected to form a system, and interoperable systems may be

interconnected to form a network. Note that this type of testing does not necessarily test the internal functioning of the individual device. If a device performs a unique function, interoperability testing would not be possible for that function.

Another form of interoperability testing is used to verify that a device (i.e., the device-under-test) appears to be working properly. If another device that performs the same, or complimentary, functions (i.e., the assured-baseline device) has itself been tested and verified to operate correctly, then it has achieved some level of assurance, and the device-under-test may be tested to verify that it interoperates with the assured-baseline device. For example, a key establishment device could be tested against another such device that is believed to operate correctly. If the two devices agree on the established key, then the test is passed. This testing produces stronger results when the device-under-test and the trusted device are independently designed and built by different organizations or individuals working independently of those associated with the device-under-test. This is because two devices built by the same group may interoperate incorrectly, but consistently, with each other. The NIST Cryptographic Algorithm Validation Program (CAVP) performs similar interoperability testing on implementations of NIST-approved cryptographic algorithms.

If a CKMS claims interoperability with another system, then the CKMS design **shall** specify the tests performed to verify the claim.

9.4 Self-Testing

A device may be designed, built, and operated correctly when first fielded, but fail some time later. Devices that fail extensively can usually be detected and replaced, but undetected failures can have major security implications. A CKMS should use devices that test themselves for integrity and security. [FIPS 140-2] specifies several self-tests to help ensure the proper operation of a cryptographic module including all its security functions.

The CKMS design **shall** specify all known self-tests and the associated CKMS functions that they verify.

9.5 Scalability Testing

Scalability testing involves testing a device or system to learn how it reacts when the number of transactions to be processed over a given period of time increases dramatically. Every device has its limitations, but some device designs may scale better than others. This is particularly true of subsystems that were not designed for modular scalability, as adding additional modules may not be feasible. In addition, subtle problems often arise that cannot be addressed by simply buying more equipment. Scalability testing is used to stress devices and systems so that these problems are known and mitigated before they become fully operational.

The CKMS design **shall** specify any scalability testing performed on the system.

9.6 Functional Testing and Security Testing

The types of tests previously described can be performed with various goals in mind. Functional testing attempts to verify that an implementation functions correctly. A functional test might determine that a cryptographic algorithm implementation correctly computes the ciphertext from the plaintext, given the key. Security testing attempts to verify that an implementation functions securely. A security test might determine that while a cryptographic algorithm implementation functions correctly (i.e. it produces the correct results), fluctuations in power consumption during the process could lead to the compromise of the key. Thus, a cryptographic algorithm implementation could pass functional testing, but fail security testing.

Penetration testing is a specific type of security testing where a team of penetration testing experts develops penetration scenarios for the system as a whole. Note that individual product/component penetration testing may be conducted as part of the product evaluation (see Section 11.1.1). The scope of penetration testing includes personnel, facilities, and procedures. The penetration team attempts to bypass the security of these components with the goal of defeating CKMS security. Any findings made by the penetration team are addressed as part of initial deployment.

The CKMS design **shall** specify the functional and security testing that was performed on the system and the results of the tests.

9.7 Limitations of Testing

Since testing is restricted to a finite number of cases that is typically significantly less than the total set of possibilities, testing does not guarantee that a device or system is correct or secure in all situations. Thus, the value of passing a test suite is directly related to the comprehensiveness and representation of the selected test cases. For example, designs often assume a particular environment (temperature, voltage, and handling) for their devices or systems. The CKMS components are then built for that environment and tested within that environment. If the device or system is used in a different environment, secure operation may be lost. Military systems are often ruggedized to handle the extreme conditions under which they may be used. This extra protection frequently comes at an additional cost.

The CKMS design **shall** specify the environment under which it is to be used.

The CKMS design **shall** specify all known conditions that are required for its secure operation.

The CKMS design **shall** specify the results of any environmental testing that was performed, including the results of stressing the system beyond its designed conditions.

10. Disaster Recovery

The use of a CKMS to protect information has the additional risk that a failure of the CKMS may hamper or prevent the usage of the information protected by the system. For example, the failure of the decrypting capability may delay or prevent the use of

encrypted data. This section describes how operational continuity may be achieved in the event of component failures or the corruption of keys and metadata.

10.1 Facility Damage

A CKMS should be located in physically secure and environmentally protected facilities. In addition, the CKMS should provide for backup and recovery in the event that damage to the CKMS occurs. The backup and recovery facilities should be designed, implemented, and operated at a level commensurate with the value and sensitivity of the data and operations being protected. When a facility is damaged, a backup facility should be brought on-line and keys that could have been disclosed accidentally should be immediately placed on Compromised Key or Certificate Revocation Lists. Wind and water damage are the common environmental risks; fire is both an environmental risk and a facility design-dependent risk.

The CKMS design **shall** specify the facility environmental, fire, and physical access control protection mechanisms and procedures for the primary and all backup facilities.

10.2 Utility Service Outage

A CKMS facility requires reliable utility services, including electricity, water, sewer, air conditioning, heat, and clean air that will assure its availability. Electrical power sufficient to satisfy the requirements of all electronic devices, human safety and comfort in normal operations and during emergencies should be provided in the primary and all backup facilities.

The CKMS design **shall** specify the minimum electrical, water, sanitary, heating, cooling, and air filtering requirements for the primary and all backup facilities.

10.3 Communication and Computation Outage

A CKMS needs sufficient communication and computation capability to perform its required functions and provide the services required by its users. Long distance communication capabilities are typically offered by commercial vendors and many computer vendors can provide computers sufficient for large key management applications. Redundant communications equipment is often installed in a CKMS to assure high availability. Remote facilities acting as hot or warm backup facilities provide even higher availability especially against environmental (e.g., weather) risks. The ability to access alternative communications services is highly desirable in the event of a communications service failure.

The CKMS design **shall** specify the minimum communications and computation redundancy needed to assure continued operation of services commensurate with the anticipated needs of users, enterprises and CKMS applications.

10.4 System hardware failure

Hardware failure can occur in any component in a CKMS. Several approaches to recover from hardware failure exist. Sufficient system redundancy can assure that the operational impact of a hardware failure is minimal, i.e., limited to reduced performance and

response time. Failed components can be repaired or replaced using locally-stockpiled spare parts. A hot backup component that automatically takes over upon detecting a primary component failure is often provided in applications requiring high availability. Users should not see any performance or response time degradation if the hot backup has the same capacity as the primary system.

Another alternative is the use of a warm backup for the primary component. A warm backup is periodically and automatically updated, but requires manual intervention to take over the role of the primary component. Aside from the duration of time for the manual cut-over, the users should not see any performance or response-time degradation.

Yet another alternative is the use of a cold backup component. A cold backup requires manual updating of its state to that of the primary component. When a primary component fails, starting the cold backup and making it operational could be time-consuming. If the cold backup is located at a different geographic location (e.g., geographic diversity may be required to protect against natural disasters), the down-time may be significant (e.g., hours or days). Once the cold backup is online, the users should not see any performance and response-time degradation if the cold backup has the same capacity as the primary component.

The CKMS design **shall** specify the strategy for backup and recovery from failures of critical hardware components.

10.5 System software failure

Software can be written to minimize software failures by using good programming practices, such as modular design and parameter bounds checking (see Appendix B of [FIPS 140-2] for additional techniques). In addition, software failures may be detected by the use of software self-tests (see Section 4.9 of [FIPS 140-2]).

Software failures may be minor, major or catastrophic in consequences. Minor errors may be due to undetected software errors (bugs) or due to temporary failures. Such errors or failures should be investigated and repaired before the CKMS is used. Major failures may be intentionally caused by corrupting the CKMS data or software. These failures should be investigated and repaired, perhaps by returning to a known secure state that was previously stored in a backup facility.

Catastrophic errors should be investigated, and a backup facility used until the primary system can be completely reloaded from a known secure state. In such situations the CKMS data created since the last secure state was saved may be lost. A CKMS should be implemented and operated under the assumption that a catastrophe will eventually occur. Therefore, it is recommended that full secure-state system backups are made on a regular basis, and that the latest CKMS secure state can be reloaded into a repaired and ready CKMS.

The CKMS design **shall** specify the strategy for backup and recovery from a major software failure.

10.6 Cryptographic module failure

Cryptographic modules should have built-in health checks that are adequate to detect hardware, software, or firmware failures. Cryptographic modules may have pre-operational, conditional, and periodic self-tests. When a failure is detected within a [FIPS 140-2]-validated module, control is passed to an error state that outputs an error indicator and determines whether the error is a recoverable or non-recoverable type of error. Sensitive data should not be output from the module while it is in the error state. If the error is non-recoverable, the module should be rebooted and pass all power-up self tests before continuing normal processing. If the non-recoverable error repeats on repeated attempts to power-up, then the module should be replaced.

The CKMS design **shall** specify what self-tests are used by each cryptographic module to detect errors and verify the integrity of the module.

The CKMS design **shall** specify how the module responds to detected errors.

The CKMS design **shall** specify its strategy for the replacement of failed cryptographic modules.

10.7 Corruption of keys and Metadata

Cryptographic keys and bound metadata may be corrupted in transmission or in storage. Corrupted keys and metadata should be replaced or corrected as soon as the corruption is detected. The replacement of corrupted keys and metadata typically involves the establishment or storage of a new key and bound metadata. If a corrupted key or a key with corrupted metadata was used to protect data, the security consequences of such an event should be evaluated, since a loss or compromise of sensitive data could result. Key establishment and key storage protocols are frequently designed to detect and replace corrupted keys.

A major disaster would imply that large numbers of operational keys and metadata were lost or corrupted beyond recovery from primary storage. If a key retrieval or key recovery system exists, then the keys and metadata could be restored. However, if the keys were not backed-up or escrowed, then they would have to be replaced with new keys and the information that the original keys protected may be lost.

The CKMS design **shall** specify its procedures for restoring or replacing corrupted stored or transmitted keys and their bound metadata.

The CKMS design **shall** specify its procedures for backing-up cryptographic keys and their bound metadata.

11. Security Assessment

CKMS security may be assessed at any time throughout its lifetime. This section highlights assessment considerations to be made during the initial deployment, during periodic (e.g., annual) reviews, and at incremental assessments after major changes. For

more information on U.S. Government security assessment practices see [SP 800-53], [SP 800-53A], [SP 800-115], and [SP 800-37-rev1].

11.1 Initial Deployment Security Assessment

Prior to deploying a CKMS, its security should be assessed. The activities that can be undertaken to assess the security of the CKMS include the following:

a) Selection of third party validated components
b) Functional and security testing of the CKMS
c) Architectural review of the system design
d) Penetration testing of the CKMS

Each of these activities are described in the following subsections.

The CKMS design **shall** specify assurance activities undertaken prior to or in conjunction with the initial CKMS deployment (i.e., define the scope of initial security assessment).

11.1.1 Third Party Validations

While there are currently no formal validation programs for the security of a CKMS, there do exist validation programs for certain components of a CKMS.

a) NIST-approved cryptographic algorithms can be validated under the NIST Cryptographic Algorithm Validation Program (CAVP).
b) Cryptographic modules conforming to [FIPS 140-2] can be validated under the NIST Cryptographic Module Validation Program (CMVP).
c) Non-cryptographic security and hardware (e.g. operating systems, DBMS, firewall, etc.) can be validated using the Common Criteria Standard (see [ISO/IEC 15408 Parts 1-3]) under the National Information Assurance Partnership (NIAP).
d) A CKMS or parts thereof could also be validated by a private entity hired by the vendor or a sponsor.

While these validation programs do not guarantee security, they can significantly increase confidence in the security and integrity of the CKMS.

The CKMS design **shall** specify any validation programs under which any of its components have been validated.

The CKMS design **shall** specify all validation certificate numbers for its validated components.

11.1.2 Architectural Review

Under this activity, a team of nationally and internationally recognized experts is assembled to evaluate the CKMS architecture. The architecture review team has access to the CKMS design information, the third party validation information, and end-to-end CKMS testing. The recommendations provided by the architecture review team are integrated in the CKMS design changes. The architecture review team also makes recommendations for penetration testing scenarios.

The architecture review team should have expertise in cryptography, cryptographic protocols, secure system design, network security, and computer security.

The CKMS design **shall** specify whether an architectural review is required before initial deployment.

If an architectural review is required, then the CKMS design **shall** specify the skill set to be covered by the architectural review team.

11.1.3 Functional and Security Testing

Testing is typically performed before initial deployment, as part of the periodic security review, and in the event of a incremental security assessment. A variety of functional and security tests may be performed by the vendor, the information owner, or a trusted third party (see Section 9).

The CKMS **shall** specify all testing that is required to be performed before initial deployment and specify the expected results.

11.2 Periodic Security Review

This review consists of an examination of the system controls, physical controls, procedural controls and personnel controls to ensure that these controls are in place as claimed. Changes to the system since the previous security review should be examined to ensure that the products/components are operating with the latest updates and security patches, and with secure configurations, and that the products maintain their third party security rating. Issues identified from the review should be addressed. In addition, periodic functional and security testing should be performed (see Section 9.6).

The CKMS design **shall** specify the periodicity of security reviews.

The CKMS design **shall** specify the scope of the security review in terms of the CKMS components.

The CKMS design **shall** specify the scope of the security review in terms of the activities undertaken for each CKMS component under review.

The CKMS design **shall** specify the functional and security testing to be performed as part of the periodic security review.

11.3 Incremental Security Assessment

When the system has undergone significant changes, it is desirable to perform an increment assessment of the changes in the following areas described in Section 11.1:

a) Changes to third party validated components since the previous security assessment
b) Architecture Review of the System Design Changes

c) Functional and security testing of the CKMS

If the cumulative system design changes are significant, an entire CKMS security assessment akin to Section 11.1 should be conducted.

The CKMS design **shall** specify the circumstances under which incremental security assessment will be conducted.

The CKMS design **shall** specify the scope of the incremental security assessment.

The CKMS design **shall** specify the circumstances under which a full security assessment will be conducted.

The CKMS design **shall** specify the scope of the full security assessment.

12. Other Considerations

The following are other considerations that may impact the design and operation of a CKMS.

12.1 Standards

Much can be learned about a CKMS by examining how it addresses applicable standards. Designs that comply with standards have the benefit of the wisdom that went into developing the standards. In addition, if the standards have validation programs that measure compliance, there is increased confidence that the CKMS has been correctly implemented. See Appendix A for a list of appropriate standards with a brief description of each.

12.1.1 Advantage of Standards

The use of approved target standards greatly increases confidence in the product or implementation. There is confidence that the standard was developed and reviewed by multiple parties working together. Complying with standards may also reduce the time-to-production for a product or time-to-operation for an implementation since the essential concepts do not have to be re-invented. Standards' conformance testing laboratories reduce errors in implementations by testing products before they are in the marketplace.

The CKMS design **shall** specify the federal, national, and international standards that are utilized by the CKMS and how conformance is tested for each.

12.1.2 Use of Conforming Products

The availability of commercial products that conform to one or more standards in a CKMS architecture greatly reduces the time and cost of producing a CKMS. The cost of a conformance-tested product is likely more than offset by the saved costs associated with designing and building a similar product without the guidance provided by appropriate standards.

The CKMS design **shall** specify which commercial products have been utilized in the design and the security standards to which they comply.

The CKMS design **shall** specify all other security standards to which the CKMS conforms.

12.2 Ease of Use

Possibly the most significant constraint to the use of a CKMS is the difficulty that some systems present to the untrained user. Since most users are not cryptographic security experts and security is only a secondary goal for them, the CKMS needs to be as transparent as possible.

12.2.1 User Perceptions, Prejudices, and Premonitions

Ease of use is very subjective. Something easy or obvious for one person may not be easy or obvious for another. Designers should keep in mind that users are not usually security experts so they may not understand the purpose of the security feature that they are operating. Security is not usually the primary purpose of the product. Past experiences, perceptions, and prejudices may taint a person's evaluation of a product. A large segment of the potential user population needs to be satisfied with a security product, including that it is easy to use, for it to be widely procured and used.

The CKMS design **shall** specify all user interfaces to the system.

The CKMS design **shall** specify the results of any tests regarding the ease of using the proposed interfaces.

12.2.2 Design Principles of the User Interface

While ease of use may be highly subjective and difficult to evaluate, several design principles for achieving this goal have been established. Ease-of-use design goals should include that:

a) It is easy to do the right thing using the CKMS (e.g., invoke a key management function).
b) It is difficult to do the wrong thing using the system.
c) It is easy to recover when a wrong thing is done.

The CKMS design **shall** specify the design principles of the user interface.

12.3 International Laws, Rules, Regulations, and Other Constraints

While communications security may be a desirable goal for most of the world, it may not be desirable for all nations. The sovereignty of a nation and its ability for self-protection are often paramount to the nation. Restricting its citizens, visitors, or un-trusted transients from obtaining certain information or services that are available in other nations can be a national goal; if not in policy, then in fact. Cryptography has historically been used primarily in military applications and predominantly during a time of war. Cost-effective, widespread secure communications and computing may neither be a goal of, nor be acceptable to, some countries. Therefore, a CKMS designer needs to be cognizant of such

potential constraints to interoperability. Either the CKMS design or its use may be constrained in some manner in order to be acceptable within such countries. Export and import controls may have some level of jurisdiction over some security systems and products.

12.4 National Laws, Rules, Regulations, and Other Constraints

Similar to a country's desire to restrict or prohibit communications security product usage within its borders for military reasons, some nations may desire to restrict such usage within their borders for law enforcement reasons. The legal authority to monitor certain telephone usage under certain conditions has long been known and accepted within many countries. The legal authority to access and use communications for law enforcement purposes within specific limits is an unresolved issue. CKMS designs should simply be cognizant of such potential constraints.

12.5 Technological Challenges

A CKMS is often intended to have a security lifetime of many years. Therefore, the designer should consider possible threats resulting from advances in technology that may render the CKMS insecure. Some examples are discussed below.

a) Attacks on Cryptographic Algorithms
   A cryptographic algorithm has an expected security life. However, as time passes new attacks may be found that reduce its security life. This in turn is likely to reduce the security lifetime of the CKMS that relies on the algorithm to protect data. Eventually, the cryptographic algorithm may need to be upgraded or replaced altogether.

   A CKMS should implement cryptographic algorithms as modules that can be replaced and updated without significantly affecting the rest of the implementation. In particular, block cipher parameters like key length and block length should be variable so that they may be increased if necessary.

   The CKMS design **shall** specify which cryptographic algorithms are used by the system.

   The CKMS design **shall** specify the expected security lifetime of each cryptographic algorithm used in the system.

   The CKMS design **shall** specify which modular components of the cryptographic algorithms are designed to be upgraded or replaced with similar, but functionally improved components.

b) Quantum Computing
   If large word size quantum computers could be built, then the security of integer factorization and discrete log-based public-key cryptographic algorithms would be threatened. This would be a major negative result for many CKMS which rely on these algorithms for the establishment of cryptographic keys. Research is

currently underway to find public-key algorithms that would be resistant to quantum computing (e.g., lattice-based public-key cryptography), but no widely accepted solution has yet been found.

The CKMS design **shall** specify the technology assumptions upon which its security is based.

c) Quantum Cryptography
Quantum cryptography is related to quantum computing technology, but viewed from a different perspective. Quantum cryptography is a possible replacement for public key algorithms that hopefully will not be susceptible to the attacks enabled by quantum computing. CKMS designers should review the technical literature on these topics before making final design decisions.

The CKMS design **shall** specify new technologies that could easily be incorporated into the system to improve security.

Appendix A: Applicable Standards and Recommendations

A short purpose statement is provided for each of the referenced items below so that the reader can immediately determine the applicability of the referenced item to the reader's needs.

1. [FIPS 140-2]
   Security Requirements for Cryptographic Modules, May 2001,
   www.csrc.nist.gov/publications/PubsFIPS.html.
   FIPS 140-2 specifies the requirements in eleven areas that must be met by cryptographic modules protecting U.S. Government information. This applies to hardware, software, firmware and hybrid modules. The standard provides four increasing, qualitative levels of security that are intended to cover a wide range of potential applications and environments. The security requirements cover areas related to the secure design and implementation of a cryptographic module. These areas include the cryptographic module specification; the cryptographic module ports and interfaces; the roles, services, and authentication mechanism; the finite state model; the physical security; the operational environment; cryptographic key management; electromagnetic interference/electromagnetic compatibility (EMI/EMC); self-tests; design assurance; and mitigation of other attacks. Compliance to this standard is validated under the NIST Cryptographic Module Validation Program (CMVP).

2. [FIPS 180-3]
   Secure Hash Standard (SHS), October 2009,
   www.csrc.nist.gov/publications/PubsFIPS.html.
   FIPS 180-3 specifies five hash algorithms that can be used to generate digests of messages. The digests are used to detect whether messages have been changed since the digests were generated.

3. [FIPS 186-3]
   Digital Signature Standard (DSS), June 2009,
   www.csrc.nist.gov/publications/PubsFIPS.html.
   FIPS 186-3 specifies algorithms for applications requiring a digital signature. It allows the use of DSA, RSA, and ECDSA signature techniques, along with an appropriate hash function from FIPS 180-3 to compute a digital signature on U.S. Government information. FIPS 186-3 also includes requirements and algorithms for the generation of keys and domain parameters. Compliance to this standard is validated under the NIST Cryptographic Algorithm Validation Program (CAVP).

4. [FIPS 197]
   Advanced Encryption Standard (AES), November 2001,
   www.csrc.nist.gov/publications/PubsFIPS.html.
   FIPS 197 specifies a symmetric key block cipher encryption/decryption algorithm for the protection of sensitive U.S. Government information. It supports key sizes of 128,

192, and 256 bits and a block size of 128 bits. Compliance to this standard is validated under the NIST Cryptographic Algorithm Validation Program (CAVP).

5. [FIPS 198-1]
The Keyed-Hash Message Authentication Code (HMAC), July 2008, www.csrc.nist.gov/publications/PubsFIPS.html.
FIPS 198-1 describes a keyed-hash message authentication code (HMAC), a mechanism for message authentication using cryptographic hash functions. HMAC can be used with any iterative Approved cryptographic hash function, in combination with a shared secret key.

6. [IPSEC]
Various IPSEC RFCs under http://www.ietf.org/dyn/wg/charter/ipsecme-charter.html.
The various IPSEC RFCs describe how authentication, encryption, and integrity security services are provided for the IP packets. The RFCs cover the format of the security services payload for the packets, various cipher suites for the security services, and key management techniques for the cryptographic algorithms used to provide the security services.

7. [ISO/IEC 15408 Parts 1-3]
Information technology – Security techniques – Evaluation criteria for IT security,
Part 1: Introduction and general model
Part 2: Security functional requirements
Part 3: Security assurance components
http://www.iso.org/iso/catalogue.
ISO/IEC 15408-1:2009 establishes the general concepts and principles of IT security evaluation and specifies the general model of evaluation given by various parts of ISO/IEC 15408 which in its entirety is meant to be used as the basis for evaluation of security properties of IT products.
ISO/IEC 15408-2:2005 defines the required structure and content of security functional components for the purpose of security evaluation. It includes a catalogue of functional components that will meet the common security functionality requirements of many IT products and systems.
ISO/IEC 15408-3:2008 defines the assurance requirements of the evaluation criteria. It includes the evaluation assurance levels that define a scale for measuring assurance for component targets of evaluation (TOEs), the composed assurance packages that define a scale for measuring assurance for composed TOEs, the individual assurance components from which the assurance levels and packages are composed, and the criteria for evaluation of protection profiles and security targets.

8. [KERBEROS]
Various Kerberos RFCs under http://www.ietf.org/dyn/wg/charter/krb-wg-charter.html.
The various KERBEROS RFCs describe how KERBEROS authentication, encryption, and ticket granting security services are provided. The RFCs cover the format of the security services payload, various cipher suites for the security services,

and KERBEROS based on password based authentication and based on X.509 certificate based authentication.

9. [RFC 2560]
Online Certificate Status Protocol (OCSP), http://www.ietf.org/rfc/rfc2560.txt.
RFC 2560 specifies a protocol useful in determining the current status of a digital certificate without requiring CRLs.

10. [RFC 3852]
Cryptographic Message Syntax (CMS), http://www.ietf.org/rfc/rfc3852.txt.
RFC 3852 describes the Cryptographic Message Syntax (CMS) format. The format consists of describing how information related to cryptographic operations such as digital signature, secure hash, authentication information, and encryption related information is carried for the arbitrary data.

11. [RFC 4251]
The Secure Shell (SSH) protocol Architecture, http://www.ietf.org/rfc/rfc4251.txt.
RFC 4251 is a protocol for secure remote login and other secure network services over an insecure network. This document describes the architecture of the SSH protocol, as well as the notation and terminology used in SSH protocol documents. It also discusses the SSH algorithm naming system that allows local extensions. The SSH protocol consists of three major components: The Transport Layer Protocol provides server authentication, confidentiality, and integrity with perfect forward secrecy. The User Authentication Protocol authenticates the client to the server. The Connection Protocol multiplexes the encrypted tunnel into several logical channels. Details of these protocols are described in separate documents.

12. [RFC 5272]
Certificate Management over CMS (CMC), http://www.ietf.org/rfc/rfc5272.txt.
RFC 5272 provides protocol standard for using certificate management services such as enrollment, rekey, and revocation using PKCS #`10 or Cryptographic Message Syntax (CMS).

13. [RFC 5273]
Certificate Management over CMS (CMC): Transport Protocols,
http://www.ietf.org/rfc/rfc5273.txt.
RFC 5273 defines a number of transport mechanisms that are used to move CMC (Certificate Management over CMS (Cryptographic Message Syntax)) messages. The transport mechanisms described are: HTTP, file, mail, and TCP.

14. [RFC 5280]
Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, http://www.ietf.org/rfc/rfc5280.txt.
RFC 5280 defines the formats for X.509 public key certificates and associated CRLs. This RFC also defined the certificates, certification paths, and CRLs processing rules.

15. [RFC 5295]
Specification for the Derivation of Root Keys from an Extended Master Session Key (EMSK), http://www.ietf.org/rfc/rfc5295.txt.
The Extensible Authentication Protocol (EAP) defined the Extended Master Session Key (EMSK) generation. RFC 5295 defines how EMSK is used to derive root keys. Root keys are master keys that can be used for multiple security services such as authentication, integrity, and encryption.

16. [RFC 5480]
Elliptic Curve Cryptography Subject Public Key Information, http://www.ietf.org/rfc/rfc5480.txt.
RFC 5480 defines the format and structure of elliptic curve related public key and signatures.

17. [SP 800-37-rev1]
Guide for Applying the Risk Management Framework to Federal Information Systems – A Security Life Cycle Approach, Final Public Draft, November 2009, http://www.csrc.nist.gov/publications/PubsSPs.html.
SP 800-37-rev1 continues the evolution to a unified framework by transforming the traditional Certification and Accreditation (C&A) process into the six-step Risk Management Framework (RMF). The revised process emphasizes: (i) building information security capabilities into federal information systems through the application of state-of-the-practice management, operational, and technical security controls; (ii) maintaining awareness of the security state of information systems on an ongoing basis though enhanced monitoring processes; and (iii) providing essential information to senior leaders to facilitate credible decisions regarding the acceptance of risk to organizational operations and assets, individuals, other organizations, and the Nation arising from the operation and use of information systems.

18. [SP 800-38A]
Recommendation for Block Cipher Modes of Operation - Methods and Techniques, December 2001, http://www.csrc.nist.gov/publications/PubsSPs.html.
SP 800-38A defines five confidentiality modes of operation for use with an underlying symmetric key block cipher algorithm: Electronic Codebook (ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB), Output Feedback (OFB), and Counter (CTR). SP 800-38A is used with an underlying block cipher algorithm that is approved in a Federal Information Processing Standard (FIPS), these modes can provide cryptographic protection for sensitive, but unclassified, computer data. Compliance to this Recommendation is validated under the NIST Cryptographic Algorithm Validation Program (CAVP).

19. [SP 800-38B]
Recommendation for Block Cipher Modes of Operation - the CMAC mode for Authentication, May 2005, http://www.csrc.nist.gov/publications/PubsSPs.html.
SP 800-38B specifies a message authentication code (MAC) algorithm based on a symmetric key block cipher. This block cipher-based MAC algorithm, called CMAC, may be used to provide assurance of the authenticity and, hence, the integrity of

binary data. Compliance to this Recommendation is validated under the NIST Cryptographic Algorithm Validation Program (CAVP).

20. [SP 800-38C]
Recommendation for Block Cipher Modes of Operation: the CCM Mode for Authentication and Confidentiality, May 2004,
http://www.csrc.nist.gov/publications/PubsSPs.html.
SP 800-38C defines a mode of operation, called CCM, for a symmetric key block cipher algorithm. CCM may be used to provide assurance of the confidentiality and the authenticity of computer data by combining the techniques of the Counter (CTR) mode and the Cipher Block Chaining-Message Authentication Code (CBC-MAC) algorithm. Compliance to this Recommendation is validated under the NIST Cryptographic Algorithm Validation Program (CAVP).

21. [SP 800-38D]
Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC, November 2007,
http://www.csrc.nist.gov/publications/PubsSPs.html.
SP 800-38D specifies the Galois/Counter Mode (GCM), an algorithm for authenticated encryption with associated data, and its specialization, GMAC, for generating a message authentication code (MAC) on data that is not encrypted. GCM and GMAC are modes of operation for an underlying approved symmetric key block cipher. Compliance to this Recommendation is validated under the NIST Cryptographic Algorithm Validation Program (CAVP).

22. [SP 800-38E]
DRAFT Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode for Confidentiality on Block-Oriented Storage Devices, August 2008,
http://www.csrc.nist.gov/publications/PubsSPs.html.
SP 800-38E approves the XTS-AES mode of the AES algorithm by reference to IEEE Std 1619-2007, subject to one additional requirement, as an option for protecting the confidentiality of data on block-oriented storage devices. The mode does not provide authentication of the data or its source.

23. [SP 800-53]
Recommended Security Controls for Federal Information Systems, February 2005,
http://www.csrc.nist.gov/publications/PubsSPs.html.
SP 800-53 provides guidelines for selecting and specifying security controls for information systems supporting the executive agencies of the federal government. The guidelines apply to all components[5] of an information system that process, store, or transmit federal information. The guidelines have been developed to help achieve more secure information systems within the federal government by:

a) Facilitating a more consistent, comparable, and repeatable approach for selecting and specifying security controls for information systems;

b) Providing a recommendation for minimum security controls for information systems categorized in accordance with Federal Information Processing Standards (FIPS) 199, *Standards for Security Categorization of Federal Information and Information Systems*;

c) Promoting a dynamic, extensible catalog of security controls for information systems to meet the demands of changing requirements and technologies; and

d) Creating a foundation for the development of assessment methods and procedures for determining security control effectiveness.

24. [SP 800-53A]
Guide for Assessing Security Controls in Federal Information Systems – Building Effective Security Assessment Plans, July 2008, http://www.csrc.nist.gov/publications/PubsSPs.html.
SP 800-53A is written to facilitate security control assessments conducted within an effective risk management framework. The assessment results provide organizational officials:

a) Evidence about the effectiveness of security controls in organizational information systems;

b) An indication of the quality of the risk management processes employed within the organization; and

c) Information about the strengths and weaknesses of information systems which are supporting critical federal missions and applications in a global environment of sophisticated threats.

25. [SP 800-56A]
Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography (Revised), March 2007, http://www.csrc.nist.gov/publications/PubsSPs.html.
SP 800-56A specifies key establishment schemes using discrete logarithm cryptography, based on standards developed by the Accredited Standards Committee (ASC) X9, Inc.: ANS X9.42 (Agreement of Symmetric Keys Using Discrete Logarithm Cryptography) and ANS X9.63 (Key Agreement and Key Transport Using Elliptic Curve Cryptography).

26. [SP 800-56B]
Recommendation for Pair-Wise Key Establishment Using Integer Factorization Cryptography, August 2009, http://www.csrc.nist.gov/publications/PubsSPs.html.
SP 800-56B specifies key establishment schemes using integer factorization cryptography, based on ANS X9.44, Key Establishment using Integer Factorization Cryptography, which was developed by the Accredited Standards Committee (ASC) X9, Inc.

27. [SP 800-57-part1]
Recommendation for Key Management – Part 1: General (Revised, March 2007, http://www.csrc.nist.gov/publications/PubsSPs.html

SP 800-57 – Part 1 focuses on issues involving the management of cryptographic keys: their generation, use, and eventual destruction. Related topics, such as algorithm selection and appropriate key size, cryptographic policy, and cryptographic module selection, are also included.

28. [SP 800-57-part3]
Recommendation for Key Management – Part 3: Application Specific Key Management Guidance, TBD,
http://www.csrc.nist.gov/publications/PubsSPs.html
SP 800-57-part 3 is intended primarily to help system administrators and system installers adequately secure applications based on product availability and organizational needs and to support organizational decisions about future procurements. The guide also provides information for end users regarding application options left under their control in normal use of the application. Recommendations are given for a select set of applications.

29. [SP 800-67]
Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher, May 2008, http://www.csrc.nist.gov/publications/PubsSPs.html.
SP 800-67 specifies the Triple Data Encryption Algorithm (TDEA), including its primary component cryptographic engine, the Data Encryption Algorithm (DEA). When implemented in an SP 800-38 series-compliant mode of operation and in a FIPS 140-2 compliant cryptographic module, TDEA may be used by Federal organizations to protect sensitive unclassified data. Protection of data during transmission or while in storage may be necessary to maintain the confidentiality and integrity of the information represented by the data. This recommendation precisely defines the mathematical steps required to cryptographically protect data using TDEA and to subsequently process such protected data. Compliance to this Recommendation is validated under the NIST Cryptographic Algorithm Validation Program (CAVP).

30. [SP 800-89]
Recommendation for Obtaining Assurances for Digital Signature Applications, November 2006,
http://www.csrc.nist.gov/publications/PubsSPs.html.
SP 800-89 specifies methods for obtaining the assurances necessary for valid digital signatures: assurance of domain parameter validity, assurance of public key validity, assurance that the key pair owner actually possesses the private key, and assurance of the identity of the key pair owner.

31. [SP 800-90]
Recommendation for Random Number Generation Using Deterministic Random Bit Generators (Revised), March 2007,
http://www.csrc.nist.gov/publications/PubsSPs.html.
SP 800-90 specifies mechanisms for the generation of random bits using deterministic methods. The methods provided are based on either hash functions, block cipher

algorithms or number theoretic problems. Compliance to this Recommendation is validated under the NIST Cryptographic Algorithm Validation Program (CAVP).

32. [SP 800-115]
Technical Guide to Information Security Testing and Assessment, September 2008, http://www.csrc.nist.gov/publications/PubsSPs.html.
This document is a guide to the basic technical aspects of conducting information security assessments. It presents technical testing and examination methods and techniques that an organization might use as part of an assessment, and offers insights to assessors on their execution and the potential impact they may have on systems and networks. For an assessment to be successful and have a positive impact on the security posture of a system (and ultimately the entire organization), elements beyond the execution of testing and examination must support the technical process. Suggestions for these activities—including a robust planning process, root cause analysis, and tailored reporting—are also presented in this guide.

33. [TLS]
Various Transport Layer Security Related RFCs under http://www.ietf.org/dyn/wg/charter/tls-charter.html.
The various TLS RFCs describe how authentication, encryption, and integrity security services are provided for the HTTP packets. The RFCs cover the format of the security services payload for the packets, various cipher suites for the security services, and key management techniques for the cryptographic algorithms used to provide the security services.

34. [X.509]
Information technology – Open Systems Interconnection – The Directory: Public-key and attribute certificate frameworks, IEC 9594-8.
This International Standard defines the formats for X.509 public key certificates and associated CRLs. This International Standard also defined the certificates, certification paths, and CRLs processing rules. This International Standard defines the formats for X.509 attribute certificates and associated CRLs. This International Standard also defined the attribute certificates, certification paths, and CRLs processing rules.

35. [XML DSIG]
XML Signature Syntax and Processing (Second Edition), W3C Recommendation 10 June 2008, http://www.w3.org/TR/xmldsig-core.
XML DSIG describes the format for digital signatures on XML documents. The Standard also describes the format for ancillary information (e.g., certificates, CRL, Signer Identifiers, etc.) that can be used to assist in digital signature verification.

36. [XML ENC]
XML Encryption Syntax and Processing, W3C Recommendation 10 December 2002, http://www.w3.org/TR/xmlenc-core.

XML ENC describes the format for encrypted XML documents. The Standard also describes the format for ancillary information (e.g., certificates, CRL, Recipient Identifiers, etc.) that can be used to assist in decryption.

Appendix C: Glossary of Terms

The following glossary contains the primary terms and definitions used in this document. Readers should also review the glossaries contained in [SP 800-57-part1].

Active State	The key lifecycle state in which a cryptographic key is available for use for a set of applications, algorithms, and security entities.
Algorithm Transition	The processes and procedures used to replace one cryptographic algorithm with another.
Application	A computer program designed and operated to achieve a set of goals or provide a set of services.
Archive	To place an electronic cryptographic key into a long-term electronic storage medium which will be maintained even if the storage technology changes. Also, the location where archived keys are stored.
Audit	The procedures performed by an audit administrator to collect, analyze, and summarize the data required in a report to the system administrator regarding the security of the system.
Backup	The process of placing at least one copy of a key in a safe facility or facilities so that the key can be quickly retrieved if the original key is lost or modified.
Binding	A cryptographic operation that links two or more data elements such that the data elements cannot be modified or replaced without being detected.
Bound Metadata	Metadata associated with a key and protected by the CKMS against unauthorized modification and disclosure.
CKMS	A set of components that is designed to protect, manage, and distribute cryptographic keys and bound metadata.
CKMS Component (Component)	Any mechanism (including hardware, software, or firmware), policy and procedures that are used to implement a CKMS.
CKMS Profile	A document that provides an implementation independent specification of CKMS security requirements for use by a community of interest (e.g., U.S. Government; banking, aerospace etc.).
Commercial Off-The-Shelf (COTS)	A product that has been designed and built to serve a large market by implementing popular components and providing popular services.
Compromise	The unauthorized disclosure, modification, substitution or use of sensitive data (e.g., keys, key metadata, and other security-related information) and loss of, or unauthorized intrusion into, an entity containing sensitive data and the conversion of a trusted entity to an adversary.
Compromised State	A key lifecycle state in which a key is designated as compromised and not used to apply cryptographic protection to data. Under certain circumstances, the key may be used to process already protected data.
Conformance	Satisfying the requirements of a specification or standard, often verified by a testing.

Cryptographic Boundary	An explicitly defined perimeter that establishes the boundary of all components of a cryptographic module.
Cryptographic Key (Key)	A string of bits, integers, or characters that constitute a parameter to a cryptographic algorithm. Some keys must be kept secret from unauthorized parties while other keys may be made public.
Cryptographic Key Management System	A system for the management (e.g., generation, distribution, storage, backup, recovery, use, revocation, and destruction) of cryptographic keys and their bound metadata.
Cryptographic Module (Module)	A set of hardware, software and/or firmware that implements security functions (e.g. cryptographic algorithms and key establishment) and encompasses the cryptographic boundary.
Cryptographic Officer	An individual authorized to perform cryptographic initialization and management functions on the cryptographic components of a CKMS.
Cryptography	Historically meant "secret writing" and used primarily for protecting secret military information; now is the science of transforming information: to a form that protects the information from unauthorized disclosure, modification, or replacement and supports authentication of the identity of the source of the information.
Deactivated State	The key lifecycle state in which a key is not to be used to apply cryptographic protection to data. Under certain circumstances, the key may be used to process already protected data.
Designer	The person or organization having the ability and responsibility and authority for specifying the components of a new system and how the components will be structured, coordinated, and operated.
Destroyed State	A key lifecycle state that zeroizes a key so that it cannot be recovered and it cannot be used. For record purposed, the identifier and other selected metadata of a key may be retained.
Destroyed Compromised State	A key lifecycle state or that zeroizes a key so that it cannot be recovered and it cannot be used and marks it as compromised, or that marks a destroyed key as compromised. For record purposed, the identifier and other selected metadata of a key may be retained.
Entity	An individual (person), organization, device or process. An entity has an identifier to which it may be bound.
Escrow	To place an electronic cryptographic key and rules for its retrieval into a storage medium maintained by a trusted third party.
Extensibility	A measure of the ease of increasing the capability of a system.
Firewall	The process integrated with a computer operating system that detects and prevents undesirable applications and remote users from accessing or performing operations on a secure computer; security domains are established which require authorization to enter.
Formal Language	A language whose syntax (i.e., rules for creating correct sentences with proper structure) is defined such that the rules are unambiguous and all syntactically correct sentences of the language can be recognized as being correct by an automaton (e.g., a computer running a syntax analysis application program).

Framework	A description of the components (i.e., building blocks) that can be combined or used in various ways to create a "system" (e.g., building, automobile, computer, CKMS).
Garbled	The modification of a cryptographic key in which one or more of its elements (e.g., bit, digit, character) has been changed or destroyed.
Generation	The key and metadata management function used to compute or create a cryptographic key.
Hash value	The fixed-length bit string produced by a hash function
Identifier	A text string used by the CKMS to select a specific key from a collection of keys.
Intellectual Property	An asset of a person or organization having value because of their creativity (e.g., copyright, trademark, patent, trade secret).
Interoperability	A measure of the ability of one set of entities to physically connect to and logically communicate with another set of entities.
Key	See cryptographic key.
Key Agreement	A key establishment procedure where resultant keying material is a function of information contributed by two or more participants, so that no party can predetermine the value of the keying material independent of the other party's contribution.
Key Confirmation	A procedure to provide assurance to one party (the key confirmation recipient) that another party (the key confirmation provider) actually possesses the correct secret keying material and/or shared secret.
Key Entry	The process by which a key (and perhaps its associated Meta-data) is entered into a cryptographic module in preparation for active use.
Key Establishment	The process by which a key is securely shared between two or more security entities, either by transporting a key from one entity to another (key transport) or deriving a key from information shared by the entities (key agreement).
Key Label	A key label is a text string that provides a human-readable and perhaps machine-readable set of descriptors for the key. Examples of key labels include: "Root CA Private Key 2009-29"; "Maintenance Secret Key 2005."
Key Lifecycle State	One of the set of finite states that describes the accepted use of a cryptographic key at that time in its lifetime including: Pre-Activation; Active; Suspended; Deactivated Revoked; Compromised; Destroyed; Destroyed Compromised.
Key Output	The process by which a key (and perhaps its bound metadata) are extracted from a cryptographic module (usually for remote storage).
Key Owner	An entity (e.g., person, group, organization, device, module) authorized to use a cryptographic key or key pair and whose identity is associated with a cryptographic key or key pair.
Key State Transition	The process of moving from one key lifecycle state to another.
Key Transport	A key establishment procedure whereby one party (the sender) selects and encrypts the keying material and then distributes the material to another party (the receiver).

Key Wrapping	A method of encrypting keys (along with associated integrity information) that provides both confidentiality and integrity protection using a symmetric key.
Malware	Software designed and operated by an adversary to violate the security of a computer (includes, spyware, virus programs, root kits, Trojan horses)
Metadata	Information used to describe specific characteristics, constraints, acceptable uses, and parameters of another data item (a cryptographic key in this document).
Meta-Language	A language used to define the formal syntax and semantics of another language (generally a new language for computer applications).
Mode of Operation	A set of rules for operating on data with a cryptographic algorithm and a key; often includes feeding all or part of the output of the algorithm back into the input of the algorithm, either with or without additional data being processed. Examples are: Cipher Feedback; Output Feedback; Cipher Block Chaining.
Parameters	Specific variables and their values used with a cryptographic algorithm to compute outputs useful to achieve specific security goals.
Pre-Activation State	A key lifecycle state in which a key has not yet been authorized for use.
Recover	To reconstruct a damaged or destroyed key after an accident or abnormal circumstance or to obtain an electronic cryptographic key from a trusted third party after satisfying the rules for retrieval.
Registration	The collection of procedures performed by a registration agent for verifying the identity and authorizations of a security entity (individual, group, device, system, organization, enterprise) and binding the entity's identifier to keys and metadata in a CKMS.
Rekey	The process used to replace a previously active key with a new key that was created completely independently of the old key.
Renewal	The process used to extend the validity period of a key so that it can be used for an additional time period.
Retrieval	To obtain an electronic cryptographic key from active or archival electronic storage, a backup facility, or an archive under normal operational circumstances.
Revoked State	The key lifecycle state in which a currently active cryptographic key is not to be used to encode, encrypt, or sign again within a domain or context.
Role	The set of acceptable functions, services, and tasks that a person or organization may perform within an environment or context.
Router	A physical or logical entity that receives and transmits data packets or establishes logical connections among a diverse set of communicating entities (usually supports both hardwired and wireless communication devices simultaneously).
Scalability	A measure of the ease of changing the capability of a system.

Scheme	A (cryptographic) scheme consists of an unambiguous specification of a set of transformations that are capable of providing a (cryptographic) service when properly implemented and maintained. A scheme is a higher level construct than a primitive and a lower level construct than a protocol.
Security Policy	The rules and requirements established by an organization governing the acceptable use of its information and services, and the level and means for protecting the confidentiality, integrity, and avaliability of its information.
Security Strength	A number associated with the amount of work (that is, the base 2 logarithm of the number of operations) that is required to break a cryptographic algorithm or system.
Semantics	The intended meaning of acceptable sentences of a language.
Store	To place an electronic data into a storage medium which may be accessed and retrieved under normal operational circumstances by authorized entities.
Suspended State	The key lifecycle state used to temporarily remove a previously active key from that status but making provisions for later returning the key to active status, if appropriate.
Syntax	The rules for constructing acceptable sentences of a language.
Transport	The process used to move a cryptographic key from one protected domain to another (includes both physical and electronic methods of movement)
Trust	A characteristic of an entity (e.g., person, process, key, or algorithm) that indicates its ability to perform certain functions or services correctly, fairly, and impartially, and that the entity and its identity are genuine.
Trust Anchor	One or more trusted public keys that exist at the base of a tree of trust or as the strongest link on a chain of trust and upon which a Public Key Infrastructure is constructed in a CKMS.
Trust Anchor Store	The location where trust anchors are stored.
Update	The process used to replace a previously active key with a new key that is related to the old key.
User	An individual authorized by an organization and its policies to use an information system, one or more of its applications, its security procedures and services, and a supporting CKMS.
Validate	The process by which cryptographic parameters (e.g., domain parameters, private keys, public keys, certificates, symmetric keys) are tested as being appropriate for use by a particular cryptographic algorithm for a specific security service and application and that they can be trusted.